

UNITED STATES AIR FORCE

ADA 248 775

SUMMER RESEARCH PROGRAM -- 1991

HIGH SCHOOL APPRENTICESHIP PROGRAM (HSAP) REPORTS

VOLUME 12

ROME LABORATORY  
ARNOLD ENGINEERING DEVELOPMENT CENTER

RESEARCH & DEVELOPMENT LABORATORIES

5800 Uplander Way

Culver City, CA 90230-6608

Program Director, RDL  
Gary Moore

Program Manager, AFOSR  
Lt. Col. Claude Cavender

Program Manager, RDL  
Claude Baum

Program Administrator, RDL  
Gwendolyn Smith

Submitted to:

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Bolling Air Force Base

Washington, D.C.

December 1991

Accession For	
NTIS	CRA&
DTIC	TAB
Unannounced	
Justification	
By	
Distribution/	
Availability	
Dist	Avail Spec
A-1	



# REPORT DOCUMENTATION PAGE

Form Approved  
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE 9 January 1992	3. REPORT TYPE AND DATES COVERED 30 Sep 90-30 Sep 91	
4. TITLE AND SUBTITLE 1991 High School Apprenticeship Program (HSAP) Volumes 10-12 Vol. 12			5. FUNDING NUMBERS F49620-90-C-0076	
6. AUTHOR(S) Mr Gary Moore				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Research Development Laboratories (RDL) 5800 Uplander Way Culver City CA 90230-6608			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFOSR/NI Bldg 410 Bolling AFB DC 20332-6448 Lt Col V. Claude Cavender			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AEOSR-TR-92 0179	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT UNLIMITED			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words)  High school students who live in communities where Air Force laboratories are located have an opportunity to spend eight weeks during the summer doing scientific research at the laboratory. Each student is assigned a mentor from the laboratory. During the summer of 1991 132 students participated in the program. Each student was required to submit a report on their accomplishments. Those student reports were consolidated and bound into this annual report.				
14. SUBJECT TERMS			15. NUMBER OF PAGES	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	

AFOSR-TR- 92 0179

Approved for public release (AFRC)  
Distribution unlimited.  
Approved for public release IAW AFR 190-12  
STINEO Program Manager

Approved for public release,  
distribution unlimited

92 4 08 008

92-09037



## PREFACE

Reports in this document are numbered consecutively beginning with number 1. Each report is paginated with the report number followed by consecutive page numbers, e.g., 1-1, 1-2, 1-3; 2-1, 2-2, 2-3.

This document is one of a set of 13 volumes describing the 1991 AFOSR Summer Research Program. The following volumes comprise the set:

<u>VOLUME</u>	<u>TITLE</u>
1	Program Management Report
<i>Summer Faculty Research Program (SFRP) Reports</i>	
2	Armstrong Laboratory, Wilford Hall Medical Center
3	Phillips Laboratory, Civil Engineering Laboratory
4	Rome Laboratory, Arnold Engineering Development Center, Frank J. Seiler Research Laboratory
5	Wright Laboratory
<i>Graduate Student Research Program (GSRP) Reports</i>	
6	Armstrong Laboratory, Wilford Hall Medical Center
7	Phillips Laboratory, Civil Engineering Laboratory
8	Rome Laboratory, Arnold Engineering Development Center, Frank J. Seiler Research Laboratory
9	Wright Laboratory
<i>High School Apprenticeship Program (HSAP) Reports</i>	
10	Armstrong Laboratory
11	Phillips Laboratory, Civil Engineering Laboratory
12	Rome Laboratory, Arnold Engineering Development Center
13	Wright Laboratory



## 1991 HIGH SCHOOL RESEARCH REPORTS

### Rome Laboratory, Arnold Engineering Development Center

<u>Report Number</u>	<u>Report Title</u>	<u>Author</u>
<u><i>Rome Laboratory</i></u>		
<b>Rome Air Development Center (RADC)</b>		
1	Pre-Commissioning and Capabilities of the Workhorse Scatterometer	Daniel Abbis
2	Integrated Services Digital Network and its Impact on Department of Defense Communications Systems	Matthew Bauder
3	The Detection of Lasing in a Diode Cavity	Andrew Gerrard
4	C-ing the Sun	Todd Gleason
5	Auger Standards for the Scanning Auger Multiprobe PHI 600	Julie Gopsill
6	RDL Research Final Report	Jason Kowalczyk
7	GEMACS Revisions and Utilization	Jason Lenio
8	Artificial Neural Networks	Sean Menge
9	Correcting a Phone Log Program for Use with Microsoft Excel	Christopher Moylan
10	An Introduction to Software Engineering and Parallel Processing	Debra Panek
11	Database Manipulation Application for use with Paradox	Thatcher Pospiech
12	Second-Generation SOCC	Jason Riordan
13	Filter/Graphing Program	Eugene Salerno
14	Prototyping on Supercard	Joseph Senus
<u><i>Arnold Engineering Development Center</i></u>		
15	Modification of Cost Estimation Programs	Kenneth Barnes
16	Direct Write Scene Generator Laser-Beam Mode Analysis	Kevin Belew
17	RTR Trailer Cable Spooling System	Mark Coleman

**Report  
Number**

**Report Title**

**Author**

**Arnold Engineering Development Center (cont.)**

18	Menu-Driven Graphical Control Input Software	Genetta Gibson
19	Remote Access to Instrument Book Information	Kenneth Harwell
20	Power Coordination Study	Heather Hopwood
21	Development of Finite Element Modeling Procedures	Kevin Johnson
22	Plume Chemistry Diagnostics	Lisa Lewis
23	Analysis of Problems Identified in Project Engineer's Notebooks in the VKF Tunnels ABC	John-Paul Motley
24	A Theoretic Technique for Assessing Combustion Instability of a Liquid Rocket Engine	Julie Reece
25	Tunnel 4T Data Tabulation and Graphs	Jonathan Sanders
26	Computer Aided Dynamic Data Monitoring and Analysis System	Jason Scott
27	High Performance Liquid Chromatography	Jennifer Trent

# **Pre-Commissioning and Capabilities of the Workhorse Scatterometer**

Daniel J. Abbis

Rome Laboratory

1991

## ABSTRACT

The OSEL Scatter Lab, performs research work in the field of stray light, or scatter, from optical surfaces. The scatter equipment consists of a Complete Angle Scatter Instrument (CASI).

This report describes capabilities, operation, and initial setup (precommissioning) of the CASI scatterometer. The types of samples that can be measured will be discussed along with a description and examples showing the various types of data output and the formats in which the data may be presented.

## ACKNOWLEDGEMENTS

We would like to thank the following people for their help in the preparation and writing of this report:

Michael Pantoliano

Dan McCurry

Lauren Coman

## SUMMARY

The Scatterometer at RADC has many parts that all work together to obtain the needed data. This report concerns itself with these parts as well as how to use them. It also shows the capabilities that it has. The report also goes as far as to tell how to take a system signature. It contains many experiments that show some of the scatterometer's capabilities.

## INTRODUCTION

Scatter is a result of surface microroughness and contamination of optical surfaces. The microroughness is a product of three sources: non-periodic grooves parallel to surface tool marks, periodic roughness of tool marks, and nearly isotropic roughness of random orientations. Contamination consists of particulates and/or molecular films on an optic's surface, and free-floating particulates located inside the telescope or near it's field of view.

The Complete Angle Scatter Instrument (CASI) at RADC was constructed by Toomay, Mathis & Assoc., Inc. (TMA). The instrument has the ability to measure: large angle scatter, and near specular scatter. CASI is able to measure forward and back scatter from near specular to close to 90° from normal. The instrument is controlled by software that gathers the scatter measurements, and calculates and plots (BRDF, BTDF).

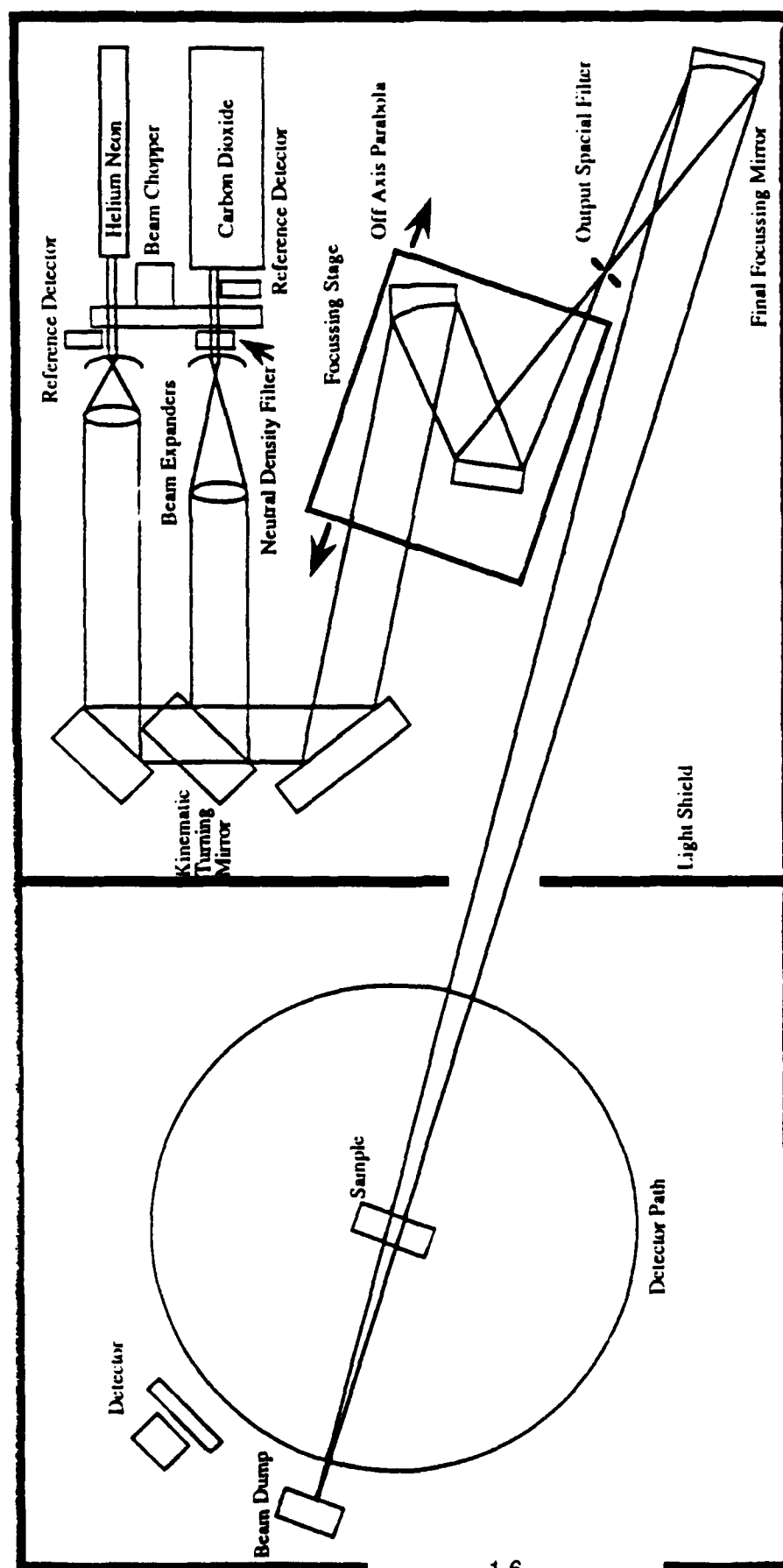


Figure 1 Scatterometer Drawing



## **1.0 Laser Classification**

There are four main laser classification groupings. These groupings are Class I, Class II, Class III, and Class IV. The Workhorse Scatterometer at RADC uses a Class IIIB Helium Neon laser and a Class IV Carbon Dioxide laser. RADC also has a 1.09 nm NdYAG laser and a 3.39 Helium Neon gas laser. To date these two lasers have not been used with regards to the Workhorse Scatterometer.

### **1.1 Introduction**

The Classification of a laser requires that the following be known: (a) the wavelength; (b) the classification duration; (c) average power output; (d) total energy per pulse; and (e) the laser source radiance. The lasers in use at RADC are both Continuous Wave (CW), therefore only the following parameters apply: (a) the wavelength; (b) average power output; and (c) the laser source radiance.

The Class IIIB Helium Neon laser has the following parameters:

- wavelength: 632.8 nm
- average power output: 7 milliwatts
- laser source radiance: 1.1 milliradians

The Class IV Carbon Dioxide laser has the following parameters:

- wavelength: 10600 nm
- average power output: 4 watts
- laser source radiance: 8-10 milliradians

### **1.2 Class III Laser Classification**

A Class III laser has the capacity to cause injury within the natural aversion response time. They can not cause serious skin damage or dangerous diffuse reflections under normal use. They must have danger labels. They are also known as "Moderate Risk" or "Medium-Power" lasers. (Sliney, David and Wolbarsht, Myron: 1980. P. 7)

#### **1.2.1 Class IIIA Laser Classification**

The Class III laser grouping is subdivided into IIIA and IIIB. Each subdivision has separate characteristics. A Class IIIA laser has an output irradiance below  $2.5 \text{ nW/cm}^2$ , and the power is below 5mW. (Sliney, David and Wolbarsht, Myron: 1980. P.596)

### **1.3 Class IV Laser Classification**

A Class IV laser may cause diffuse reflections that are dangerous to the eye and may also cause serious skin injury from direct exposure. They may cause combustion of

flammable materials. They are also known as "High Power" or "High Risk" lasers. (Sloney, David and Wolbarsht, Myron: 1980. P. 7)

## **2.0 Laser Controls**

The safety rules for the above classifications may be summarized as follows:

### **2.1 Class III Controls:**

Class III laser controls are as follows:

- a. avoid direct exposure to eye;
- b. allow only experienced personnel to operate;
- c. make an attempt at enclosing as much of the beam path as possible;
- d. place filters, shutters, and polarizers at the laser exit port to reduce the beam's power to the minimal useful level;
- e. a warning light or buzzer should indicate laser operation;
- f. operate laser in a restricted area;
- g. place the laser beam path well above or well below eye level of all present;
- h. use eye protection, and
- i. use a key switch.

### **2.2 Class IV Laser Controls**

Class IV Controls are as follows: (In addition to the Class III controls)

- a. operate within a localized enclosure;
- b. eye protection is needed for all individuals in the area, and
- c. always use beam shutters, beam polarizers, and beam filters

## **3.0 System Safety**

It is important to take a detailed look at hazard analysis and general safety procedures when planning system safety.

### **3.1 The Laser's Hazards Potentials -- System Safety**

It is possible to develop a classification system to show the need for improvement of the safety aspects in evaluating the risks that a laser and other general hazards can present. Figure 2.

### 3.1.1 Laser Hazard Analysis

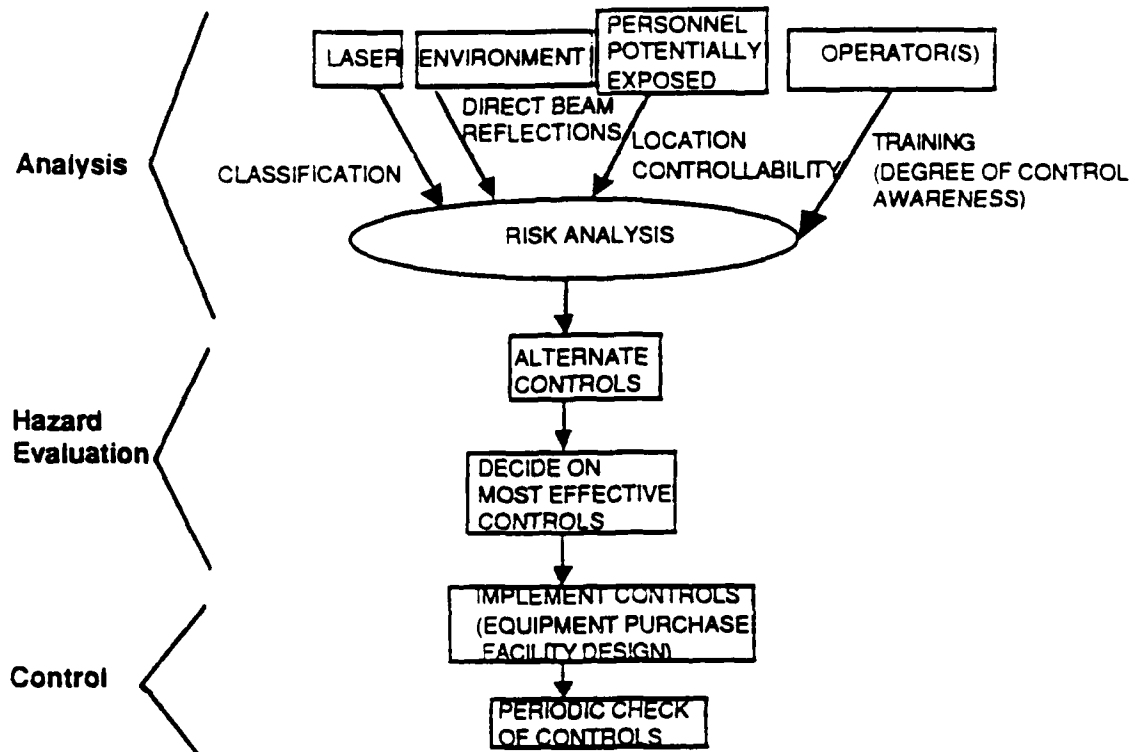


Figure 2 Laser Hazard Analysis (Sliney, David and Wolbarsht, Myron; 1980. P. 472)

### 3.1.2 General Concepts of System Safety

Regardless of the fact that the Bureau of Radiological Health Regulations assures that there must be certain safety features put into manufactured laser products, it should be recognized that these features are not in and of themselves the only means of reducing risks. The possibility of applying a greater amount of system safety controls in addition to the already existing features should be taken into consideration. Figure 3 shows a basic method used by system designers for finding possible malfunctions and safety hazards in a piece of equipment. (Sliney, David and Wolbarsht, Myron; 1980. P. 473)

## 4.0 Laser Set-Up Safety

When speaking of laser set-up safety, it is important to consider not only the laser, but also the entire laser set-up (i.e. beam paths, beam enclosures, controlled entry, etc.)

### 4.1 Laser Eye Protection

Safety goggles or spectacles are extremely effective when other engineering controls are not possible. It should be noted that the filter material and side shields should

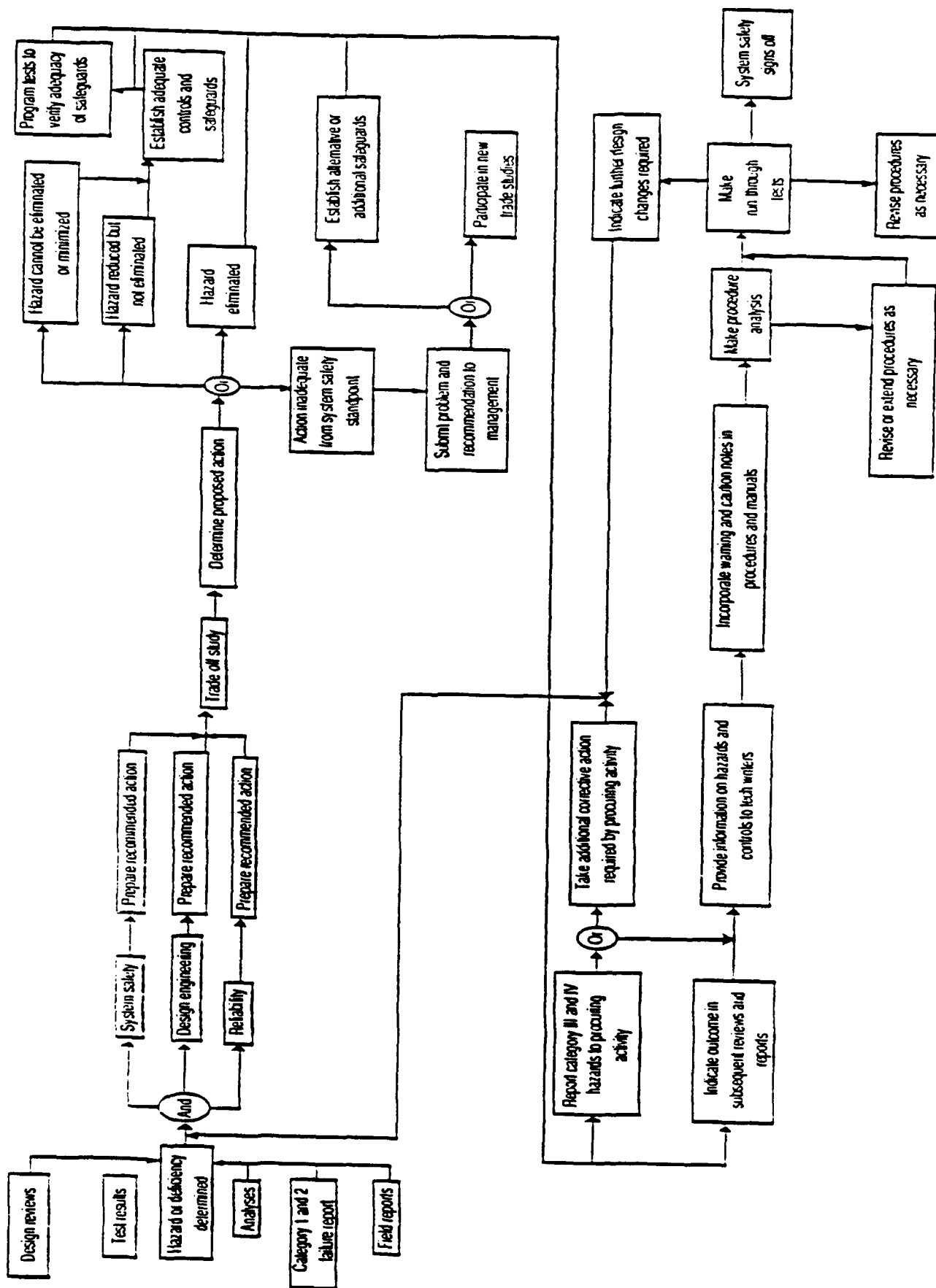


Figure 3 System Safety Flow Chart

be able to withstand the maximum irradiance encountered in the environment for at least three seconds. There are therefore many physical parameters that the goggles/spectacles must meet. Some of these parameters that must be met are: the correct protective wavelength; the correct optical density of the filter; the proper amount of visual transmittance; a high damage threshold; and the proper filter curvature. (Sloney, David and Wolbarsht, Myron; 1980. P. 10)

#### **4.2 Diffuse Reflections**

Looking directly at the beam is not the only manner in which damage to the eye can be done. Hazardous reflections are potentially dangerous to the human eye. The shaded area in Figure 4a shows the dangerous zones for intrabeam viewing of a Class 3 laser. If the output radiant exposure of a pulsed laser is increased so that the laser is considered Class 4 then the probability of eye damage is shown by the shaded area in Figure 4b.

### 4.3 Diffuse Reflections Drawing

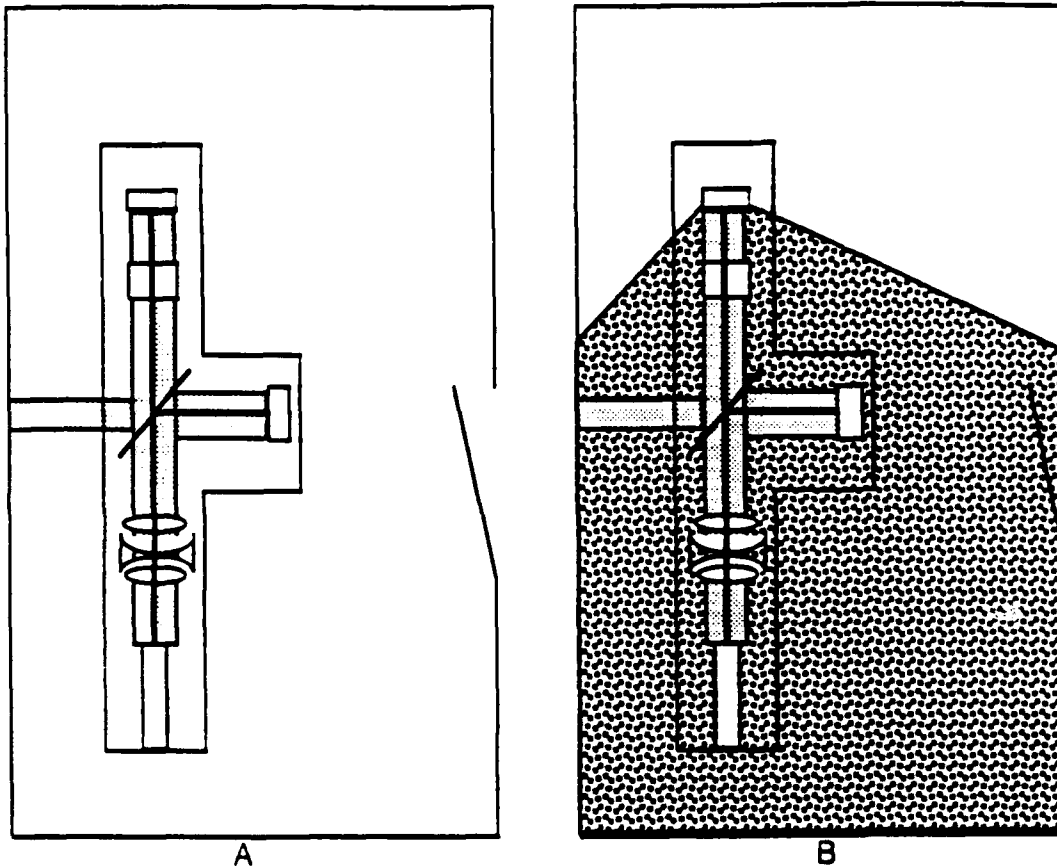
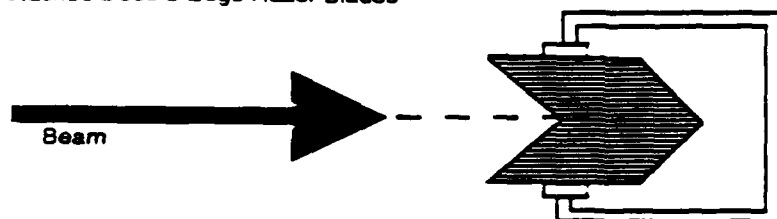


Figure 4 Diffuse Reflections. Potentially hazardous (shaded) areas in a laser laboratory are vastly different for a pulsed laser that produces only hazardous specular reflections (A), rather than hazardous diffuse reflections as well (B). (Sloney, David and Wolbarsht, Myron; 1980. P. 568)

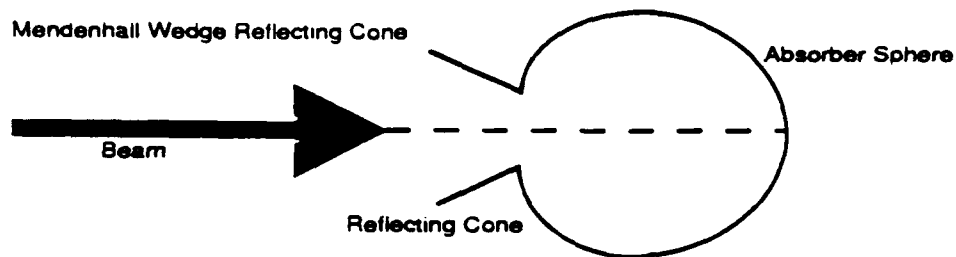
### 4.4 Beam Traps

A beam trap helps minimize stray specular reflections. It should be made of a rough, diffuse, low reflectance level material. Figure 5 shows various types of beam traps.

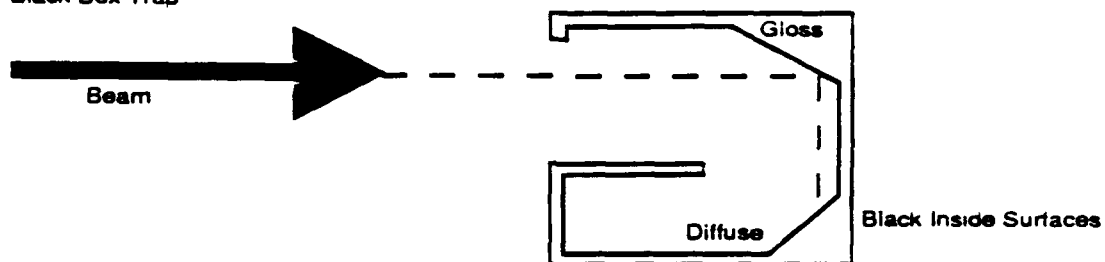
1. Stacked Double Edge Razor Blades



2. Mendenhall Wedge Reflecting Cone



3. Black Box Trap



4. Specular Reflection Box

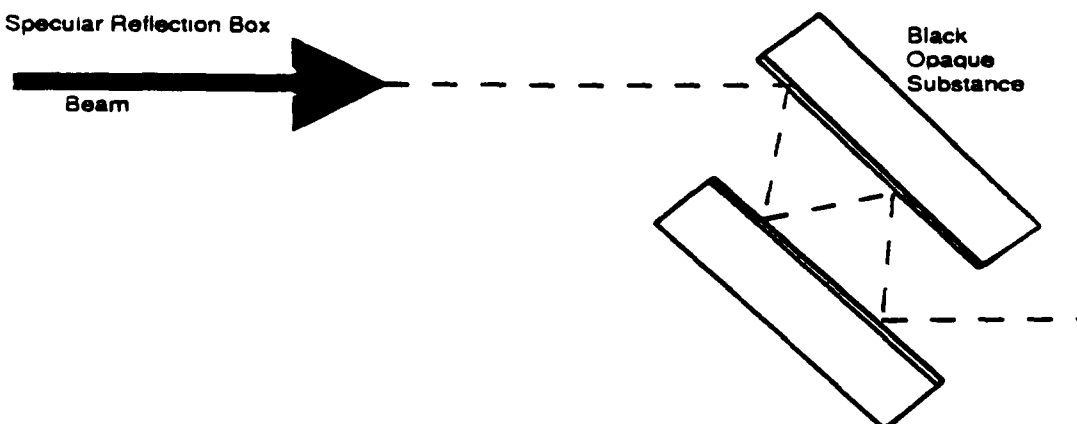


Figure 5 Beam Traps. In each of the four cases the concept is to minimize the reflected energy leaving the trap without placing too great a heat burden on the absorbing surfaces. (Slincy, David and Worbacht, Myron; 1980. P. 570.)

#### **4.6 Controlled Entry**

For Class IV lasers and certain class III it is generally advisable to limit entry. Restrictions on entry are generally done through the use of signs, special door latches, and interlocks. Class IV lasers sometimes provide a power supply relay connector to connect with a door switch to stop power to the laser when the door is opened.

#### **4.7 Beam Path**

The main hazard that a laser beam poses, besides the primary beam itself, are any secondary beams. It is extremely important that the laser beam, both primary and secondary be located above or below eye level to reduce the risk of eye damage.

#### **4.8 Temporary Beam Enclosures**

Temporary beam enclosures are extremely helpful because they allow for viewing of the beam without posing a danger by permitting body exposure to the beam. Small portable shields of glass or transparent plastic are frequently used. Such enclosures may also reduce the problem of dust collection upon optical surfaces.

#### **4.9 Standard Operating Procedure**

It is suggested that a standard operating procedure (SOP) be written. This serves many purposes. The lab personnel are forced to consider safety and generally will follow their own rules more readily than those made by a central safety officer. (Sliney, David and Wolbarsht, Myron: 1980. P. 573)

#### **5.0 RADC Scatter Laboratory Safety Checklist**

Now that the manner in which the ideal safety cautious laser laboratory should be set up has been presented, it is necessary to take a closer look at exactly what safety devices/procedures RADC has implemented in the course of the use with the Workhorse Scatterometer:

- Operating personnel: All of the Workhorse Scatterometer personnel are completely experienced in the set-up, operation, and safety of the entire system.
- Beam enclosures: RADC does not have beam enclosures to date. A major reason for this is that enclosures would only hamper the operators from being able to change such things as the detector head's apertures, tip, and tilt.



- Filters, shutters, and polarizers: RADC has all the proper and necessary laser beam filters, shutters, and polarizers to assure a safe operating environment and testing atmosphere.

- Warning lights, buzzers, and signs: RADC has all the needed safety signs and warnings.

- Restricted area: The Workhorse Scatterometer is operated in a key locked room which is located inside a combination locked area.

- Laser beam level: The laser beam is situated so that it is above the eye level of someone sitting and below the eye level of someone standing.

- Eye protection: RADC has many different types of eye protection. RADC's collection of safety glasses/goggles has a wide variety of optical density's to assure an extremely safe operating environment.

- Key switch: All the lasers used in the Workhorse Scatterometer contain a key switch.

- Beam traps: RADC uses a stacked double edge razor blade style of beam trap.

## **6.0 Signal Detector Alignment**

Now that the safety aspects have been covered, it is necessary to go through the needed pre-commissioning. Below is a step by step manner in which to get the Scatterometer ready to take a system signature.

Manually center the aperture Y axis stage. Install the 340 nm aperture plate. Adjust the detector post to center the aperture on the beam in Y. Choose the manual mode in the CASI software. Move the scan rotary axis in order to center the beam on the aperture horizontally by peaking the detector power reading. In order to fine tune the detector output it is necessary to adjust the pitch and yaw micrometers while watching the detected power displayed on the computer screen.

Open the alignment iris on the X stage assembly all the way. Use scotch tape to fix a piece of white paper to the front of the iris to make a diffuse spot. Remove the aperture plate and move the Sample  $\theta$  stage to +5 degrees from beam center. A small signal should be seen from the signal detector. Adjust the detector mount pitch and yaw micrometers to peak the signal while viewing the detector watts on the screen. Put the detector back to beam center. The detector reading on the screen should show about 2-5 mW of power and the output meter should have a positive reading. Replace the aperture and remove the paper from the iris. A signature center can now be run. (TMA (Hardware Reference Manual), April 20, 1988: pgs. 24-25)

## **7.0 System Initialization**

This section will cover the needed system alignment/initialization procedures required immediately prior to taking data

### **7.1 Prerequisites to System Initialization**

Once the system is set up and aligned the system initialization can be carried out. System initialization depends upon the optical train of the scatterometer being aligned well enough so that movement of the focusing stage does not require large adjustments to the output pinhole position.

### **7.2 Focussing the Beam**

The last adjustment after the system is aligned is focussing the specular beam at the detector plane. Using the focusing stage the output pinhole and output optics are moved with respect to the focussing mirror until a tight focused spot is obtained at the detector. Moving the output pinhole towards the focussing mirror will move the focused spot away from the mirror, while moving the output pinhole away from the focusing mirror has the opposite effect and draws the focused spot in towards the focusing mirror. The goal of this step is to position the focused spot at the front face of the detector aperture. Focus is best assessed by using a piece of film at the detector and visually inspecting the spot size and shape.

The beam is considered focused at the detector when the spot appears slightly elongated in the vertical. (TMA (Hardware Reference Manual), April 20, 1988; pgs. 26-27)

### **7.3 Rough Centering of the Detector in Y**

After the focused spot is positioned at the front plane of the detector, the detector must be moved in  $\theta$  and Y so that the focused spot is centered on the face of the detector aperture. The smallest detector aperture should be in place on the detector during centering.

The initial centering is begun by making sure that the detector aperture y stage is at its center of travel. The detector pitch and yaw stages must also be near their center of travel. The detector should be roughly aimed at the sample if this is true. Rough centering of the detector will be accomplished by adjustment the height of the detector post mount. Using the y travel adjustment on the detector post mount the height of the detector is adjusted so that the focused spot of the specular beam is at the same height as the detector aperture.

Once this is completed, the detector post mount is tightened and secured in this position. (TMA (Hardware Reference Manual), April 20, 1988; pgs. 27)

#### **7.4 Centering the Detector's Field of View**

The detector's field of view is limited to an area slightly larger than the illuminated spot on the sample. Therefore, it is necessary that the detector is aimed well at the sample.

There are two degrees of freedom of detector motion that are used to aim the detector. These motions are detector pitch (up and down) and detector yaw (side to side) motion. To adjust the pitch and yaw of the detector a dummy sample must be mounted to give the detector a bright spot at the sample location to center on. A diffuse transmissive sample is needed for the dummy sample. A piece of scotch tape stretched across an iris works well. This dummy sample must be positioned in  $z$  so that it is in the center of rotation of  $\theta_s$ .

Once the sample is in place, the detector is rotated about five degrees off the specular. Movement of the detector requires that the CASI software is running and that the operator has executed the manual mode that allows manual control of the instrument. Then remove the aperture. Now the detector's pitch and yaw is adjusted to maximize the signal the detector sees. After this, the smallest aperture, is replaced onto the detector, the detector is moved back to specular, and the diffuse transmissive sample is removed.

This procedure must be done at system installation and also each time the detector position on the sweep arm is changed or each time the height of the detector is changed by adjusting the detector post mount. It is also a good idea to periodically adjust this to insure the system is aligned.

#### **7.5 Final Centering of Detector in Sample $\theta$ and $Y$**

After the proceeding steps, the focused specular beam should be near the center of the detector aperture. the instrument should be still under manual control through the CASI software and the detector should still be in the hardware home position.

The detector is now moved by selecting the Detector Theta Axis for movement in the CASI software and stepped in  $\theta$  until the focused spot is visually centered in the detector aperture.

The scan setup in the CASI Angle Scan module should be defined with the appropriate sized aperture. The  $y$  stage on the detector aperture must also be adjusted to visually center the aperture on the focused spot. The signal measured by the detector should now approach the level of the total signal.

After this procedure is done the automatic centering routine in the Angle Scan software should be run. Once this is complete the aperture Y stage should be carefully adjusted again to maximize the detected signal. The automatic routine should be run one more time if the detector aperture was moved in y.

The only reason that the entire process needs to be done again is if the system is changed. This would include changing the detector arm length or any other system alignment positions. (TMA (Hardware Reference Manual), April 20, 1988: pgs. 28)

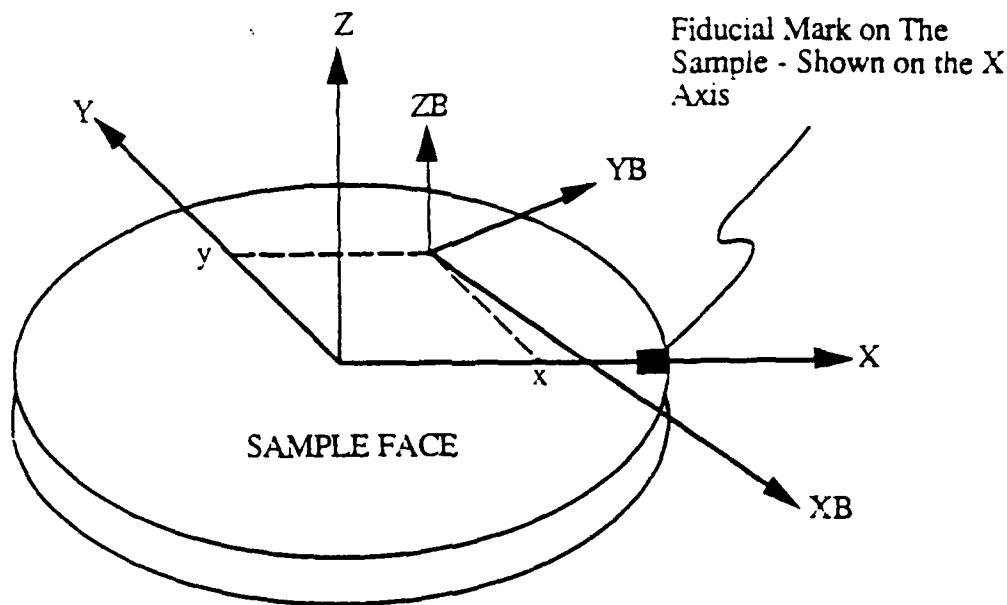
## **7.6 Measuring the Instrument Signature**

Once these initialization steps have been completed the instrument is ready to take data. The first data taken after system installation should be the Instrument Signature. The Instrument Signature is the beam profile of the instrument as measured by sweeping the detector through the focused beam.

By measuring the Instrument Signature the alignment of the instrument is checked. the background optical noise of the instrument is measured and a baseline for comparisons of scatter data is set. The Instrument Signature determines the angular range sample scatter data can be measured. It also forces the measurement of the incident power required for the computations of BSDF to be done. (TMA (Hardware Reference Manual), April 20, 1988: pgs. 28)

## **8.0 Polarization Geometry**

The relationship between the sample (X,Y,Z) and the beam (XB,YB,ZB) coordinate systems is as follows (Figure 6) the Z and ZB axes are always the local normal to the sample face. Locations on the same face are measured in the sample coordinate system. The incident and scatter directions are measured in the beam coordinate system. If the sample fiducial mark is not an X axis mark, the intended value must be indicated on the sample. (ASTM (Standard Practice for Angle Resolved Optical Scatter Measurements on Specular and Diffuse Surfaces), pg. 12)



Note 1: The X-Y zero position on the sample face is assumed to be the geometric center of the sample.

Note 2: The fiducial mark can be on the edge or back of the sample.

Figure 6 · Relationship between sample and beam coordinate systems

## 9.0 Scatter Viewed As Diffraction

An important fact is that scatter may be viewed as diffraction. Diffraction theory predicts the intensity and location of light diffracted from an input specular beam out into the sphere surrounding the diffracting component. The grating equation is a direct result of diffraction theory and is used to define the location of diffracted (scattered) light. It is useful to define the grating equation in terms of our geometry at this point.

Plane of Incidence

$$\sin \theta_s = \sin \theta_i + N \lambda / A$$

$$N = 0, \pm 1, \pm 2$$

Full Hemisphere

$$(\text{Using } N = \pm 1 \text{ and defining } 1/A = f = (f_x^2 + f_y^2)^{1/2})$$

Grating propagates in the  $\theta$  direction

$$\sin \theta_s \cos \theta_s = \sin \theta_i \pm f_x \lambda$$

$$\sin \theta_s \sin \theta_s = f_y \lambda$$

The grating equation defines diffraction (scatter) location only: so far, nothing has been said about intensity. In general, the larger surface defects are the larger the scatter will be. (Stover, John C.; 1989, P. 1-4)

## 10.0 Definition of Terms

In order to fully understand the meaning of scatter and its impact on the measuring of optical scatter it is necessary to comprehend the meaning and context of terms.

### 10.1 Total Integrated Scatter (TIS)

The instrument gathers a large fraction of the light scattered into the hemisphere in front of the sample and focuses this light onto a single detector. The detector reading is normalized by the reflected or transmitted specular light and this ratio is referred to as the "TIS".

Davies Paper was published in 1954. It concerns itself with radar. It gives the following formula for TIS:

$$TIS = \frac{\text{Reflected Scattered Light}}{\text{Reflected Specular Light}}$$

This equation is derived from:

$$TIS = 1 - e^{-\left(\frac{4\pi\delta}{\lambda}\right)^2} \approx \left(\frac{4\pi\delta}{\lambda}\right)^2 \text{ for } \left(\frac{4\pi\delta}{\lambda}\right)^2 \ll 1$$

Where  $\lambda$  is the light wavelength and  $\delta$  is the root mean square (rms) roughness. (Stover, John C.; 1989, P. 2-1)

Bennett and Porteus built an Optical TIS Scatterometer in 1961. They used Davies result. This scatterometer denotes the start of optical scatter measurement as a real source of metrology information. (Stover, John C.; 1989, P. 2-2)

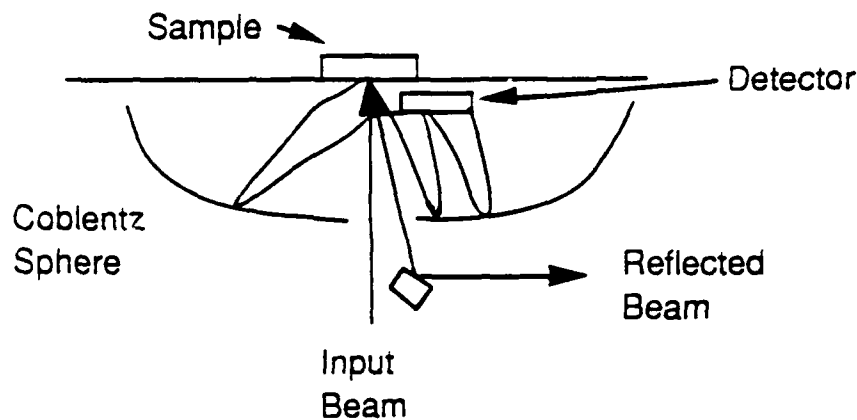


Figure 7 TIS Instrument

Advantages:

- Relatively Cheap
- Fast
- Repeatable

Problems:

- Detector reflection losses =  $f\theta_i$
- $\cos \theta_s \sim 1$  is assumed

Present Use:

- Reflection samples
- Transmission samples
- Raster scans
- $\theta_i > 0$

## 10.2 Bidirectional Distribution Functions (BSDF/BRDF/BTDF)

Measurement of light scattered as a function of an angle gives additional information. BRDF was defined in its present notation by Nicodemus in 1970 and is used to describe the scattering properties of optics. It can be defined for reflecting (BRDF) or transmitting (BTDF) optics. The BRDF is defined as

$$\text{BRDF} \equiv F \equiv \frac{dP_s / \omega_s}{P_i \cos \theta_s}$$

Where:

- $dP_s$  = power scattered in the  $\theta_s, \phi_s$  direction through  $d\omega_s$
- $d\omega_s$  = the solid angle containing  $dP_s$
- $P_i$  = the incident power

A Lambertian surface exhibits a constant BRDF. Lambertian surfaces are rough and do not meet the smooth surface criteria for TIS  $\delta$  calculations. (Stover, John C.: 1989, P. 2-3)

Thus if BRDF is known, the scattered power through any aperture,  $d\omega_s$ , is easily found for any input  $P_i$ . A slightly different form of the BRDF is often used and is referred to as "cosine corrected".

$$\text{BRDF} \equiv \frac{dP_s/\omega_s}{P_i}$$

### 10.3 Instrument Signature

Instrument signature, sometimes called instrument profile, refers to the scatter in a BRDF measurement that is caused by the instrument itself. The signature is usually confined to a region within a few degrees of specular and is caused by:

- Scatter in the instrument optics
  - Specular light reflected off the detector and back onto the sample space
  - Ghost reflections in the instrument
  - Limitations on the focussed spot size due to diffraction and aberrations
- (Stover, John C.; 1989, P. 2-7)

## 11.0 Scatterometer Components

A scatterometer is made up of many different components. In order to fully understand how a scatterometer works it is necessary to know what the parts of a scatterometer are.

### 11.1 Laser Sources and Optics

Coherent light is not required - the laser is just a very convenient light source

The first spatial filter:

- Removes scatter
- Converts beam wander to beam power variations

Chopped light / lock in electronics

Monitor beam power

Polarizers and ND Filters

Second spatial filter removes scatter

Final Focusing Element:

- Determines sample spot size
- Determines focused spot size



- Strong contributor to signature

## 11.2 Sample Holders

A sample holder can be very simple to very complex. As many as seven degrees of mechanical freedom are possible:  $\theta_i$ , fine tilt, out of plane tilt,  $\alpha$ , x, y, z.

## 11.3 Detectors (Stover, John C.; 1989, P. 3-2)

Below are listed some types and different features of some detectors.

### Types

Si	.25 $\mu$ -- 1.1 $\mu$
Ge	.5 $\mu$ -- 1.9 $\mu$
HgCdTe	2 $\mu$ -- 13 $\mu$
InAs	1 $\mu$ -- 5.5 $\mu$
PMT's	Visible/Near IR

### Important Features

- Sensitivity
- Dynamic Range
- Low Noise

### Housing

- Limited Field of View
- Programmable Low Noise Pre-Amp
- Variable Apertures
- Y Motion
- $\theta_s$  Motion
- Tilt Adjust
- BP Filter, Diffuser

## 11.4 The Computer System (Stover, John C.; 1989, P. 3-2)

The computer system is made up of many different parts. Below is a listing of most of these elements.

### Control of Motorized Axes

- "Manual" Control
- "Automatic" Control

### Control of Programmable Electronics

- Pre-Amp Gains
- Lock in Gains

- Detector Bias

#### Control of Beam Devices

- Shutters
- Turning Mirrors
- Apertures
- ND Filters

#### Control of Data Collection

- Set-up File
- Real Time Data Presentation
- Data Storage
- Data Delivery

#### Data Presentation and Analysis

- BRDF, BTDF, BSDF
- Log-Log, Log-Linear, Linear-Log, Linear-Linear
- Comparison
- Surface Statistics
- Scatter Predictions -  $\theta_i$ ,  $\lambda$
- Hard Copies

## 12.0 Experiments and Results

It is necessary to show just what the workhorse scatterometer at RADC can do through the use of completed experiments. The experiments on the following pages show some of the many capabilities that the workhorse scatterometer has.

Rome Air Development Center, Optical Scatter Laboratory

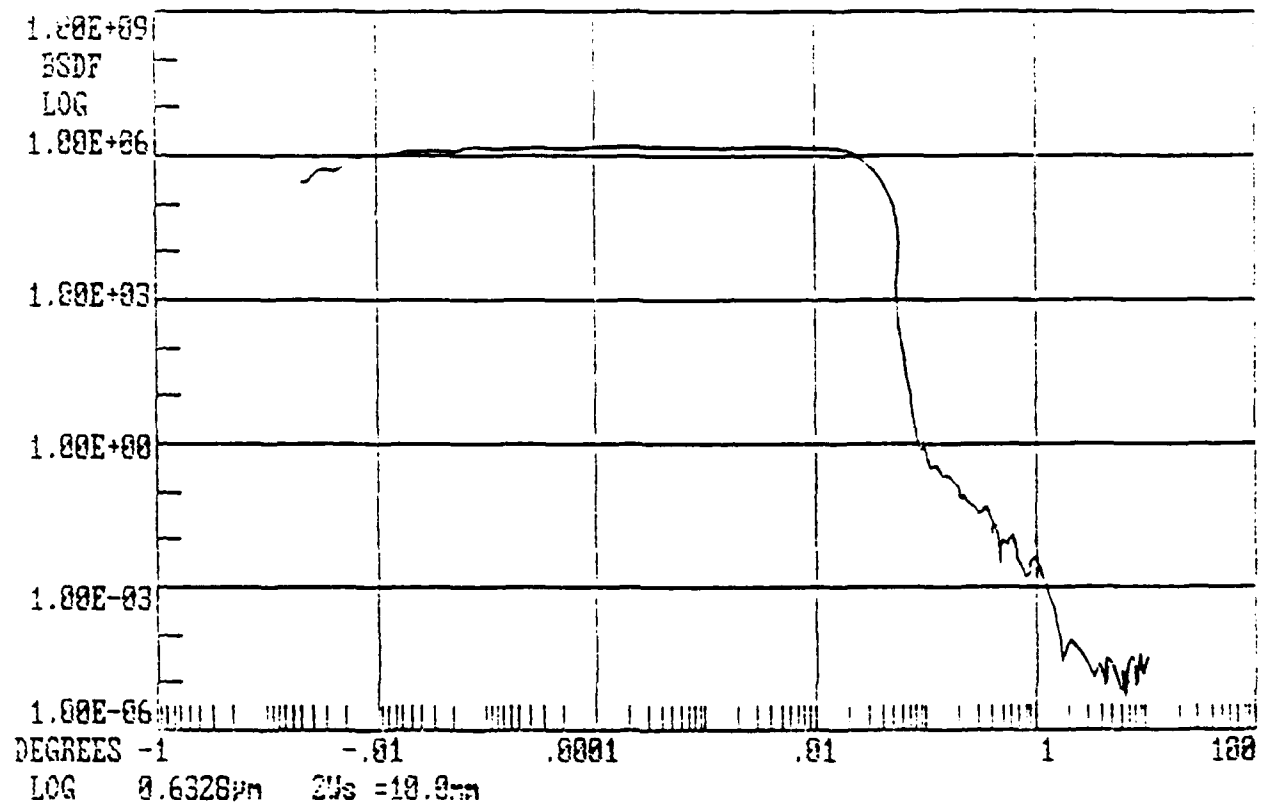


Figure 8: Normal Daily Signature

A system signature must be taken on a daily basis or before any measurements are taken. A system signature shows the amount of scatter present in the instrument for the day. This is used as the baseline for the day.

Rome Air Development Center, Optical Scatter Laboratory

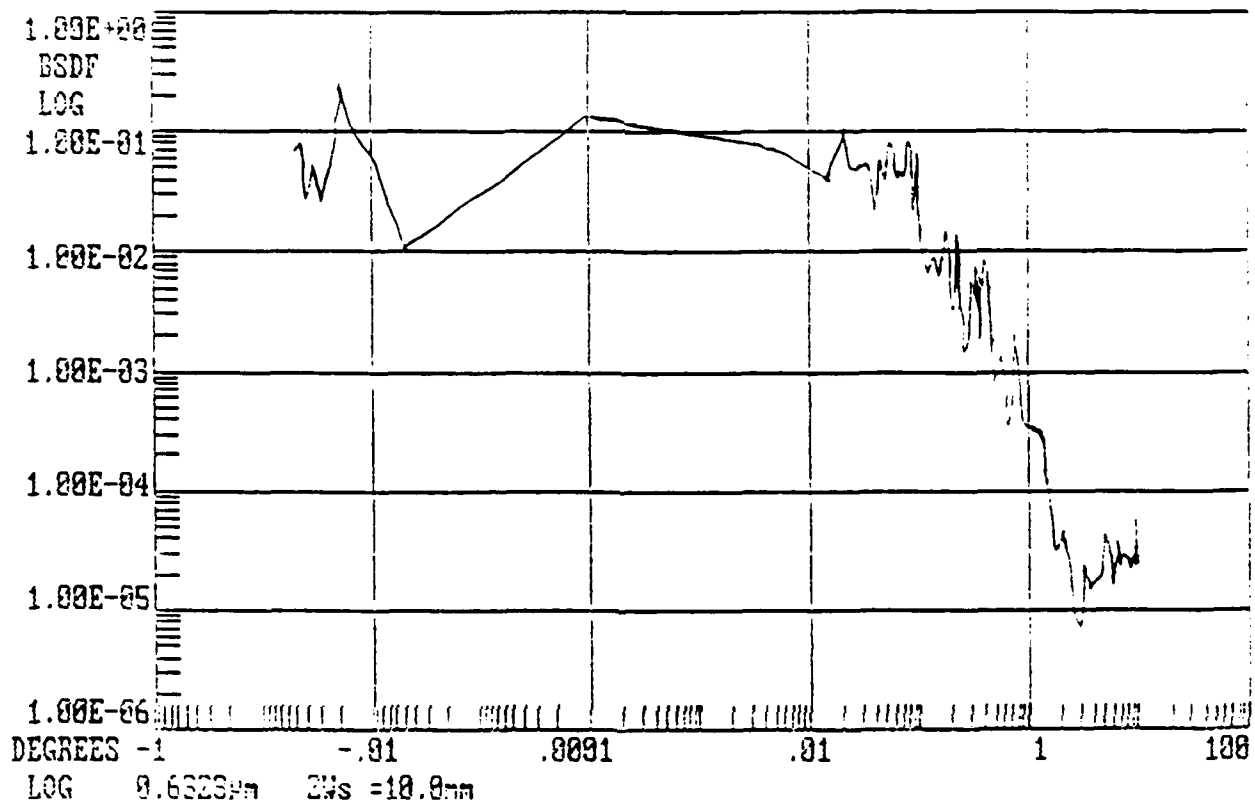


Figure 9: Signature Taken .5° Off Of Specular

Boeing Development Center, Optical Scatter Laboratory

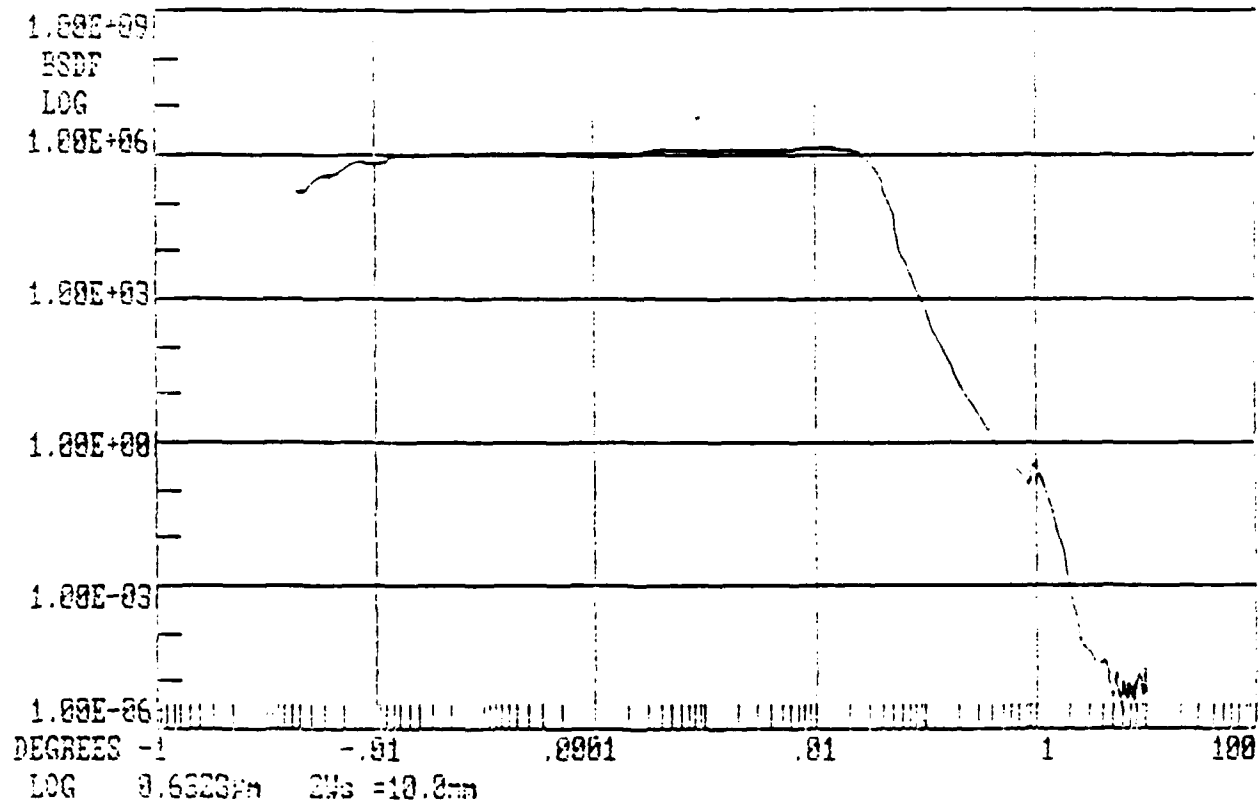


Figure 10: Signature Taken With An Iris Diameter of .54"

This graph demonstrates how an iris acts as another noise parameter in the system and thus causes the signal to fall off more rapidly.

Rock Air Development Center, Optical Scatter Laboratory

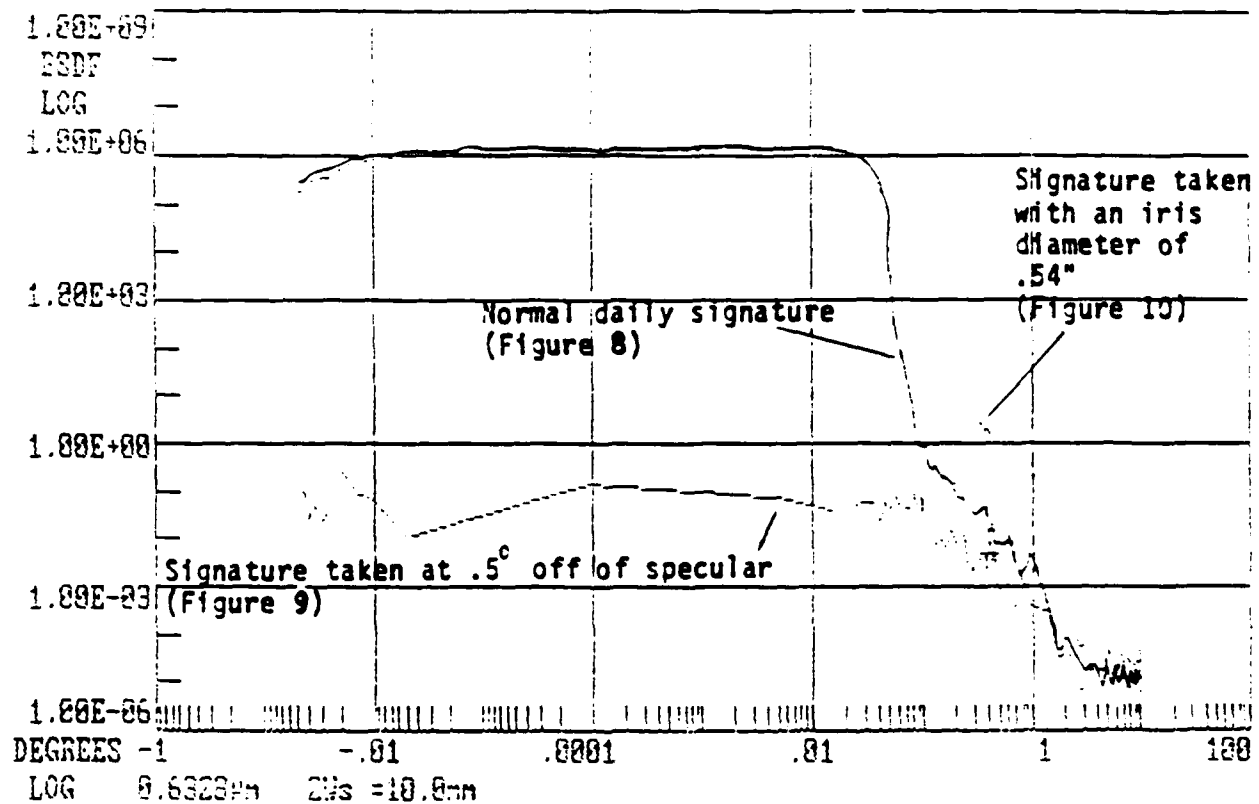


Figure 11: Comparison Graph

Rome Air Development Center, Optical Scatter Laboratory

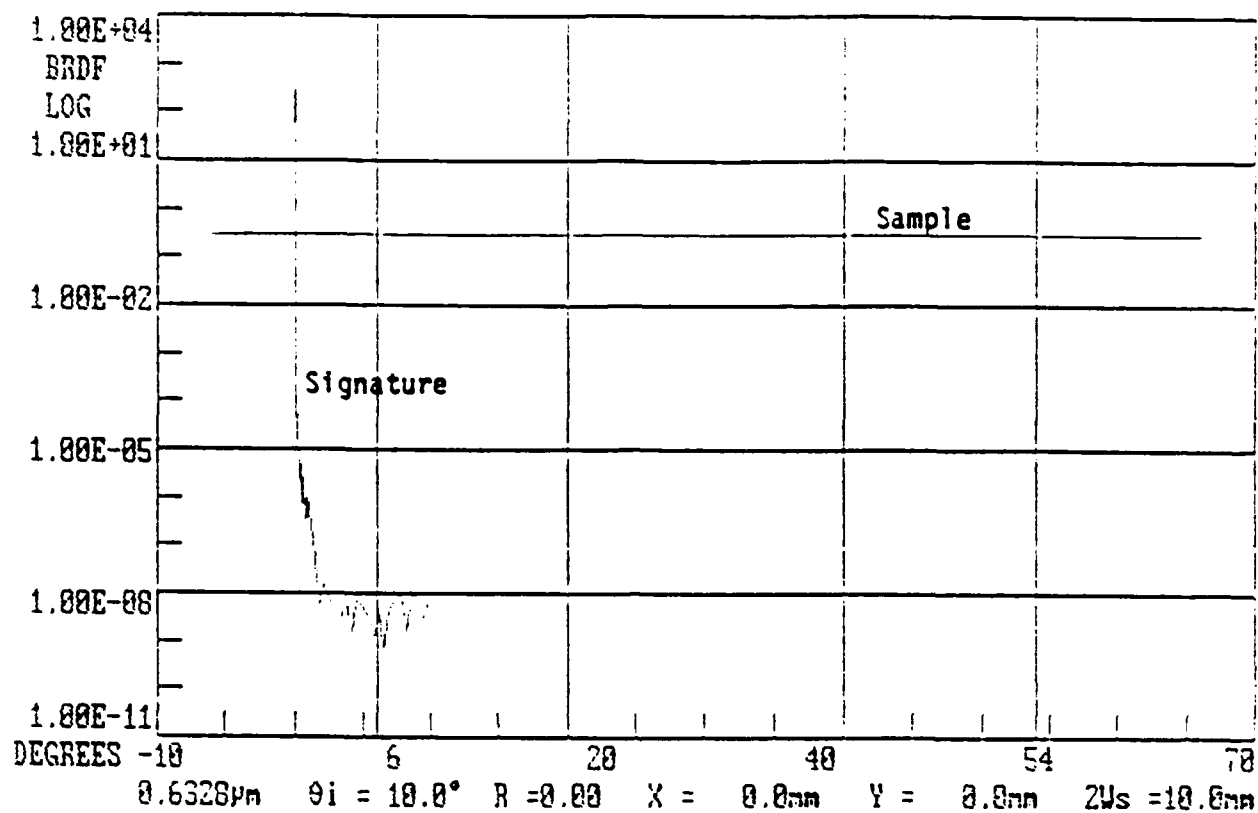


Figure 12: White Diffuse Sample Taken Near Specular

Rome Air Development Center, Optical Scatter Laboratory

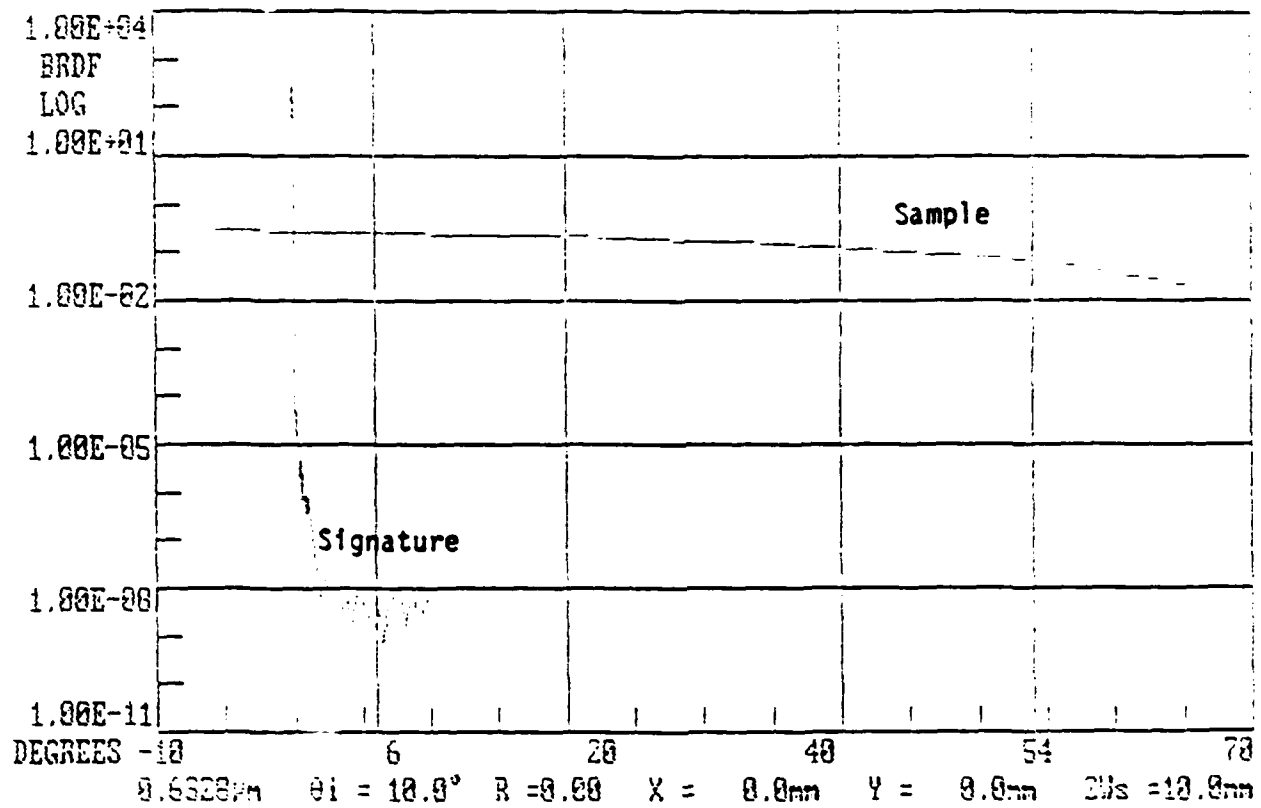


Figure 13: White Diffuse Sample Moved Forward  
From Center Of Rotation 5mm



Rome Air Development Center, Optical Scatter Laboratory

12.9 DEG 2.19E-02 BRDF

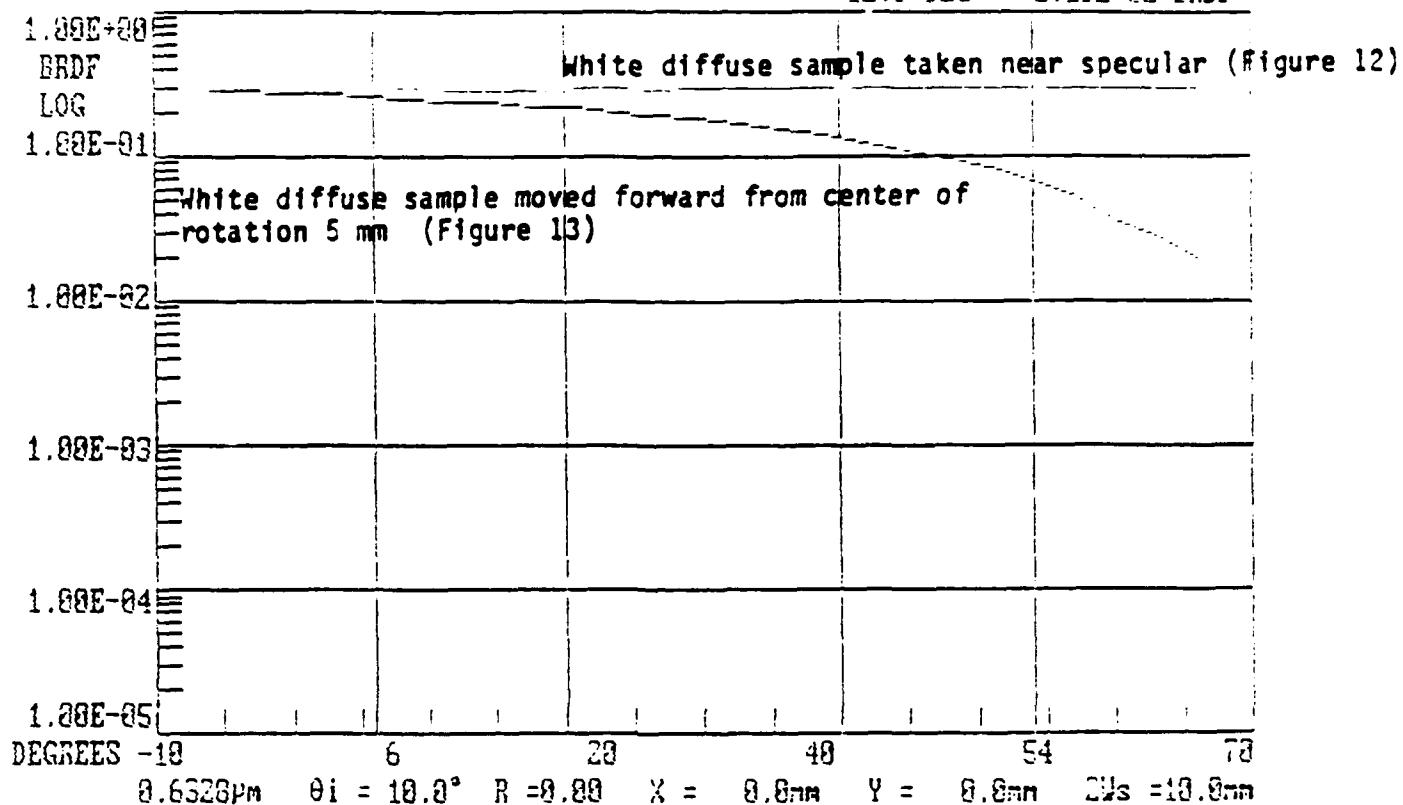


Figure 14: Comparison Graph

Home Air Development Center, Optical Scatter Laboratory

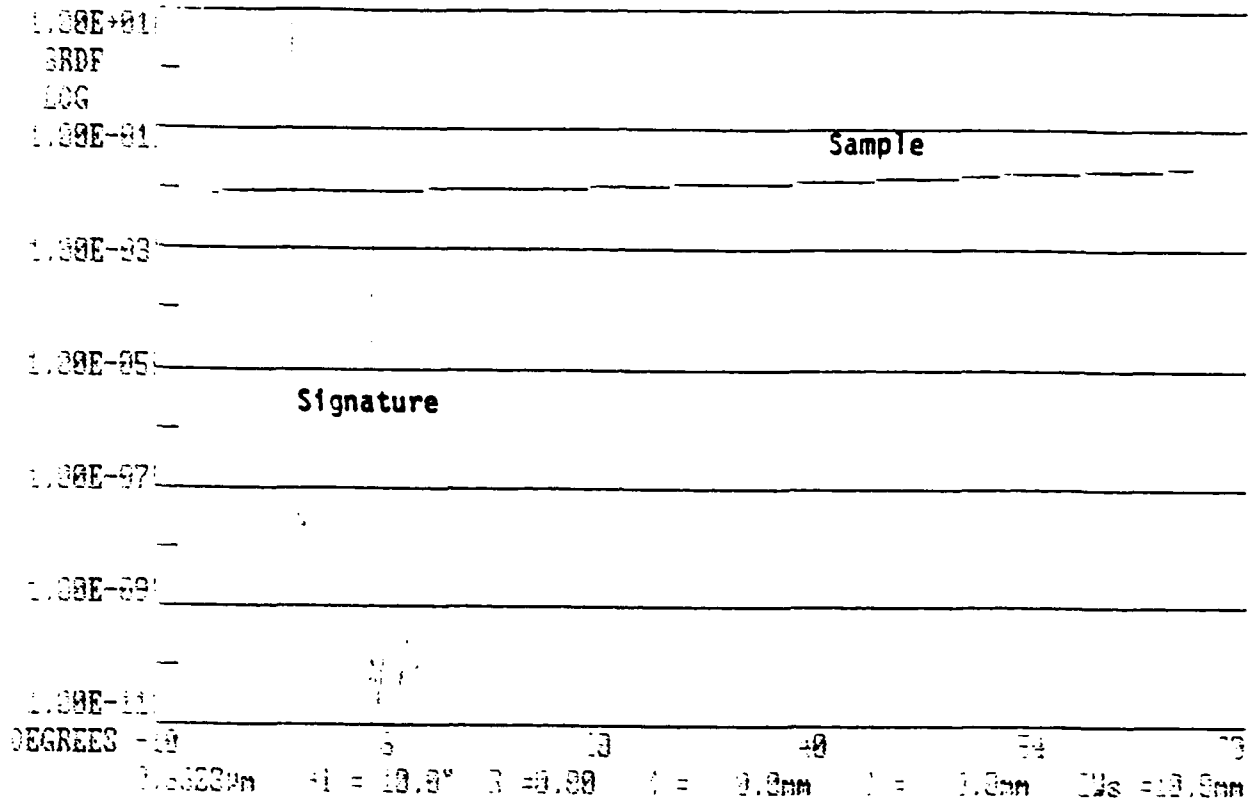


Figure 15: Black Diffuse Sample Taken Near Specular

Gene Hie Development Center ATLAS COPCO LABORATORY

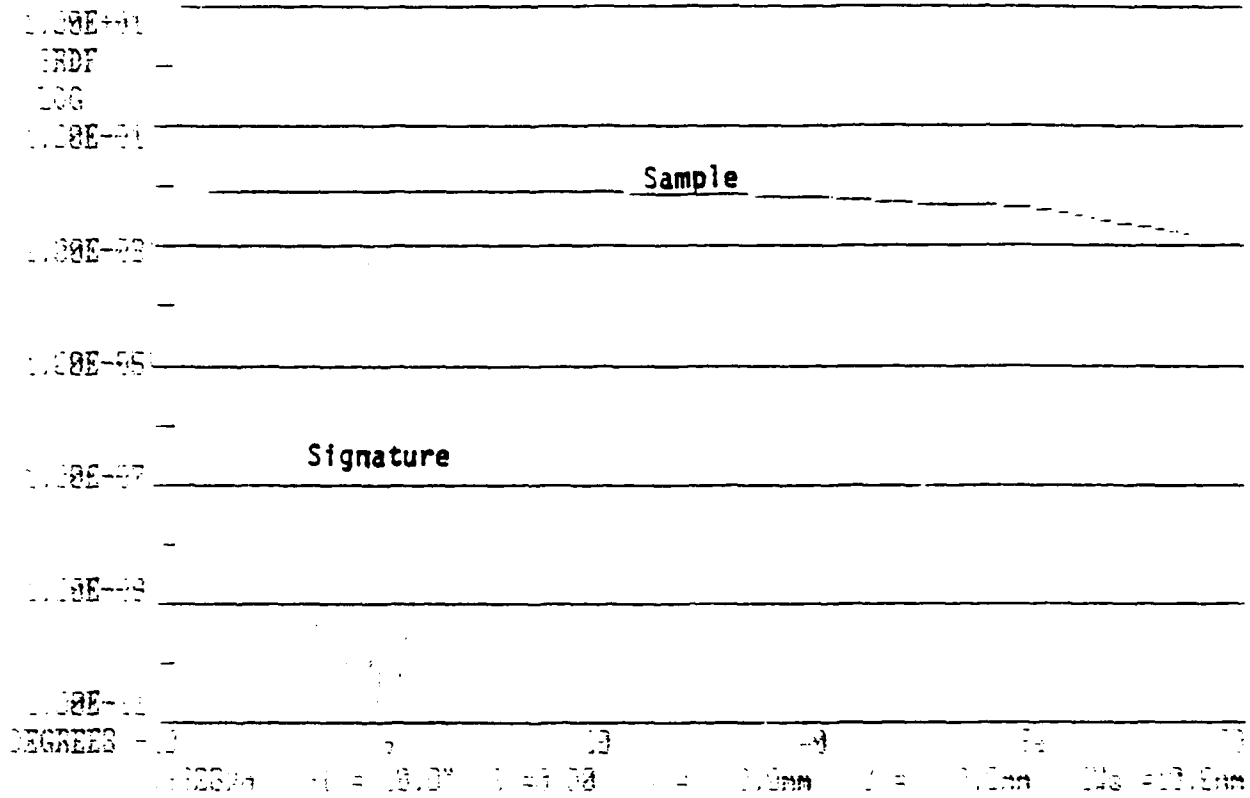


Figure 15: Black Diffuse Sample Moved Forward From Center Of Rotation 5mm

Rome Air Development Center, Optical Scatter Laboratory

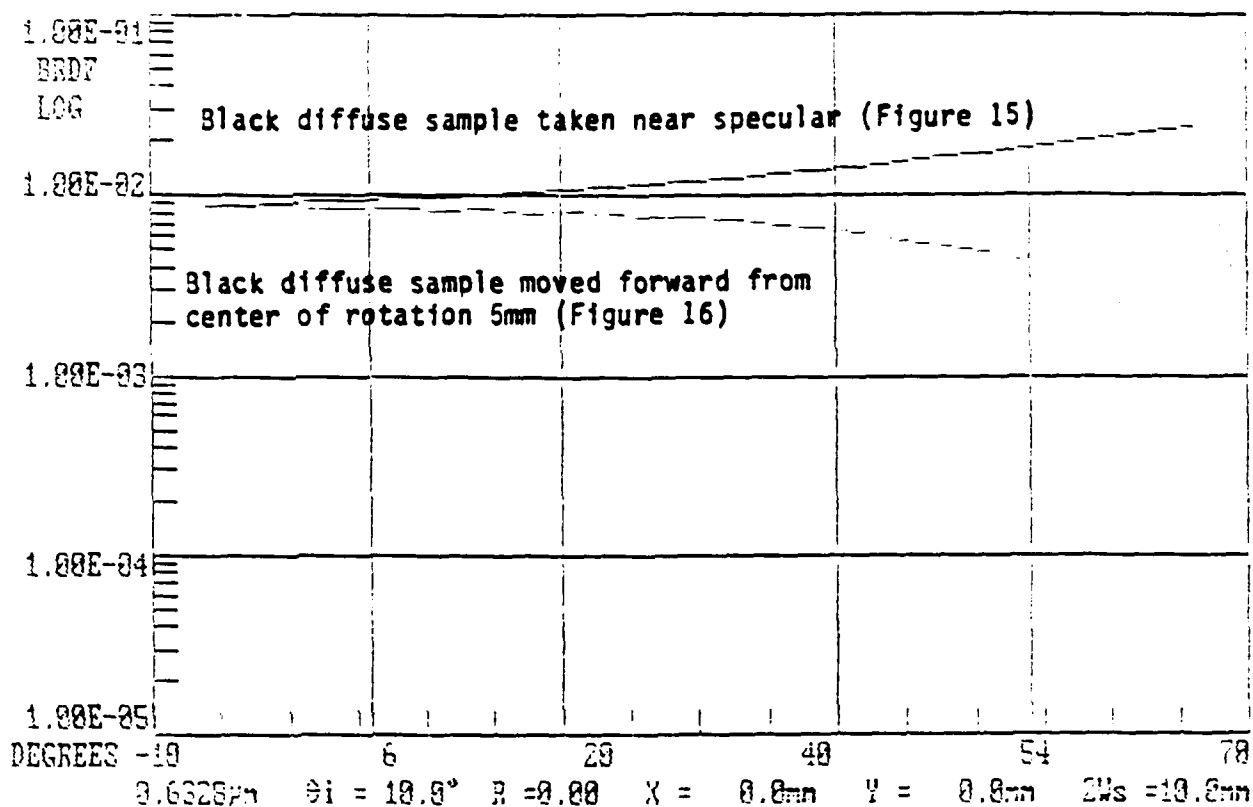


Figure 17: Comparison Graph

## RECOMMENDATIONS

After reviewing some aspects of the scatterometer and problems that it has, the writers of this paper have the following recommendations for making the scatterometer at RADC into a more precise instrument:

- Permanent beam enclosures around the table, and
- A clean room environment.

Both of these simple, but yet important aspects can eliminate a major portion of dust particles that cause scatter. Therefore, if these two simple recommendations are initiated the workhorse scatterometer at RADC will greatly become a much more precise instrument that it is presently.

## REFERENCES

Sliney, David and Wolbarsht, Myron; 1980.

TMA (Hardware Reference Manual) April 20, 1985.

ASTM (Standard Practice for Angle Resolved Optical Scatter Measurements on Specular and Diffuse Surfaces).

Stover, John C. (Optical Scatter); 1989.

# INTEGRATED SERVICES DIGITAL NETWORK AND ITS IMPACT ON DEPARTMENT OF DEFENSE COMMUNICATIONS SYSTEMS

Matthew Bauder

## ABSTRACT

Integrated Services Digital Network (ISDN) will produce major changes in the way people communicate, provide and receive services, and purchase merchandise. Business management will also see a major improvement with implementation of ISDN. The Department of Defense (DoD) can use ISDN services for military programs and for the improvement of Command, Control, Communications, and Intelligence (C<sup>3</sup>I) functions. This paper will discuss ISDN concepts and applications, and how the DoD can take advantage of this technology.

## 1.0 INTRODUCTION

ISDN is defined as a network that supports many different services, accessible by a set of standard interface devices. It will evolve from the Integrated Digital Network (IDN), which is a telephone network with all digital components. It will be built up slowly into a worldwide network and will take many years to combine all the services. Most of today's separate networks will still be around for a long time.

ISDN will become necessary in the future, because of the diversity of networks now being implemented, and the necessity to coordinate the interaction between these networks. There are so many separate networks and services in existence, that it is sometimes difficult and expensive to manage and maintain them, in addition to ensuring that they remain interoperable. ISDN will be able to consolidate and coordinate the services that are now available by many different networks.

## 2.0 ISDN

### 2.1 ISDN CONCEPTS

The ISDN backbone is an IDN, setup much like today's telephone network. It has central offices, to which the users are connected with a "digital pipe", a transmission media that has a smaller bandwidth than the IDN. Central offices are be connected by the IDN. However, the user does not have to go through the IDN if the service wanted is connected to the same central office.

The Open Systems Interconnection (OSI) Reference Model is a framework for a set of standards related to computer to computer communications. OSI is necessary for communication between different types of computers. It takes a message, and changes the data format of it and sends it over the network. Then the message is changed to the data format of the receiving computer so it can understand the message. For example, if someone using an Apple computer wants to send a message to someone who has an IBM



computer, the OSI changes the format of the data into the standard ISDN data format and sends it over the network. Then the message format is changed into IBM format and the person gets the message. It makes applications independent of the hardware they operate on.

The OSI model is divided into a 7 layer architecture, each with its own subset of functions. Layer 1, the *Physical* layer, handles the electrical and mechanical (activation, deactivation, maintenance, etc.) aspects of a connection. The second layer, the *Data Link* layer, allows for the connection to go through the *Physical* layer with the right synchronization and error flow control. The *Network* layer handles the routing and switching of information. The fourth layer, the *Transport* layer, monitors the transfer of information and makes sure the data gets delivered error free, with no losses or duplications. The *Session* layer handles user to user connection and error recovery. Layer 6, the *Presentation* layer defines the syntax of the information and interprets it. The seventh layer, the *Application* layer, allows user access to the OSI.

Figure 1 shows the OSI Reference model. If User 1 wants to communicate with User 2, the message goes first to the *Application* layer. It is then sent to the *Presentation* layer and it continues down to the physical media which passes up the *Physical* layer of User 2. From there it goes to the *Data Link* layer and all the way up the layers until the message gets to User 2. Peer protocols, sets of rules that allow two devices to communicate, are the same in the corresponding layers and are necessary for the computers to

communicate properly. When data gets to a layer, that layer completes its function and passes it on to the next layer.

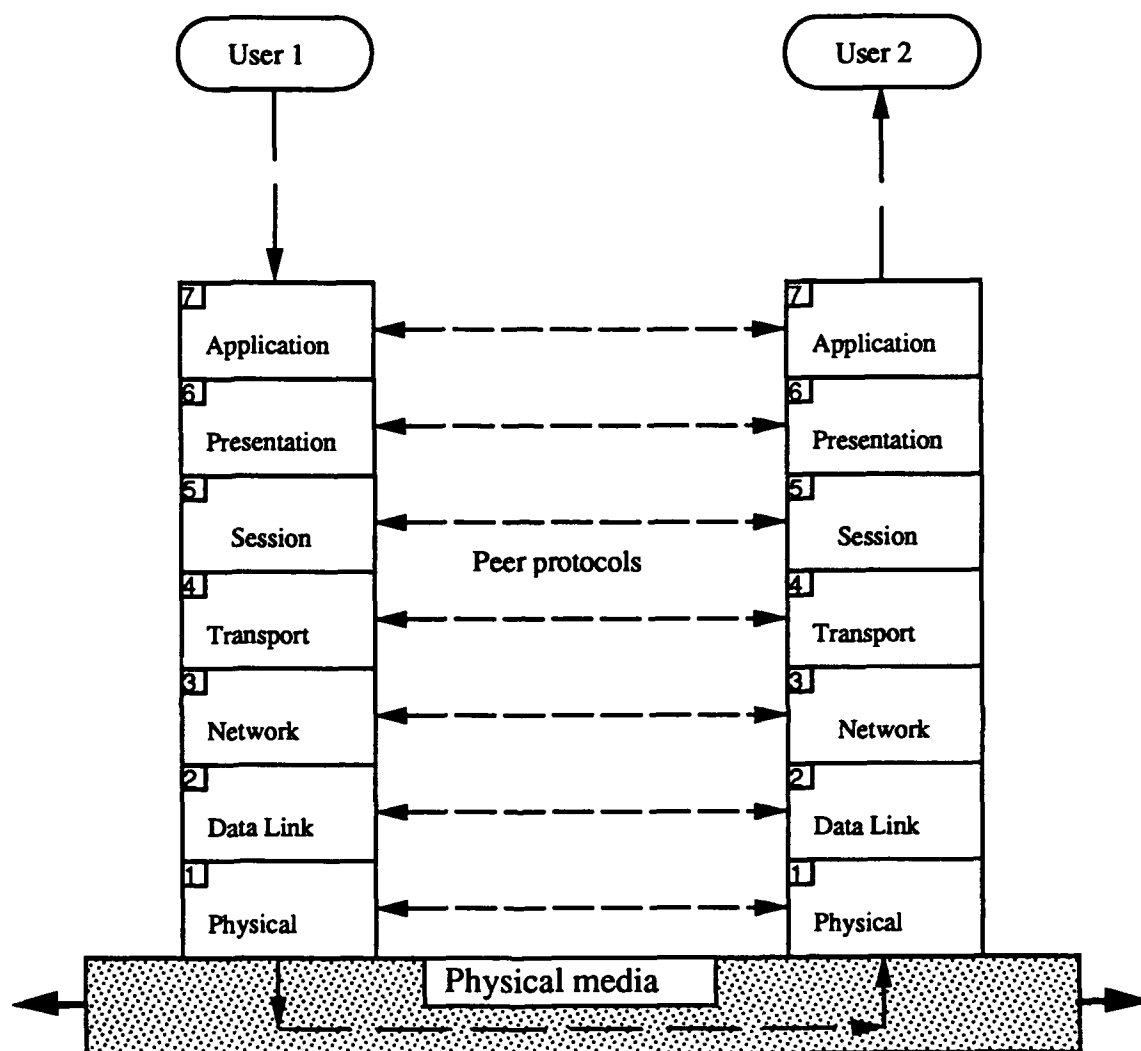


Fig. 1- The OSI Reference Model

The transmission link between the user and the ISDN central office will be divided into three channels; the B channel, the D channel, and the H channel. The B channel has a rate of 64 kbps and is used for data and digital voice. The D channel runs at 16 or 64 kbps and carries signalling information for connection over the B

channel, and transmits low-speed interactive data. The H channel is used for video and has rates of 384, 1536, and 1920 kbps.

ISDN will integrate video and data on a single network by sending them over different channels. Video will be transmitted over the H-Channel and data will travel over the B-Channel.

Two types of access to the network are provided, basic access and primary access. Basic access consists of two B channels and one 16 kbps D channel (2B+D) and uses the B channel for voice and data transmission and the D channel for signalling. Primary access is used for high capacity requirements. In the United States and Japan, it has 1.544 Mbps channels (23B+D), and Europe has 2.048 Mbps channels (30B+D). Both are provided over a single interface.

ISDN will use two different types of switching, circuit switching and packet switching. Circuit switching is where two communicating devices are physically connected through the network for the entire duration of the call. An example of this type of switching is found in today's telephone network. Circuit switching is best suited for voice communication because voice is delay sensitive and has a long hold time. (The average phone call lasts about ten minutes.) Frequency Division Multiplexing (FDM) is used with circuit switching. FDM divides the frequency of the network between users and each user has that channel for as long as needed (see Fig. 2).

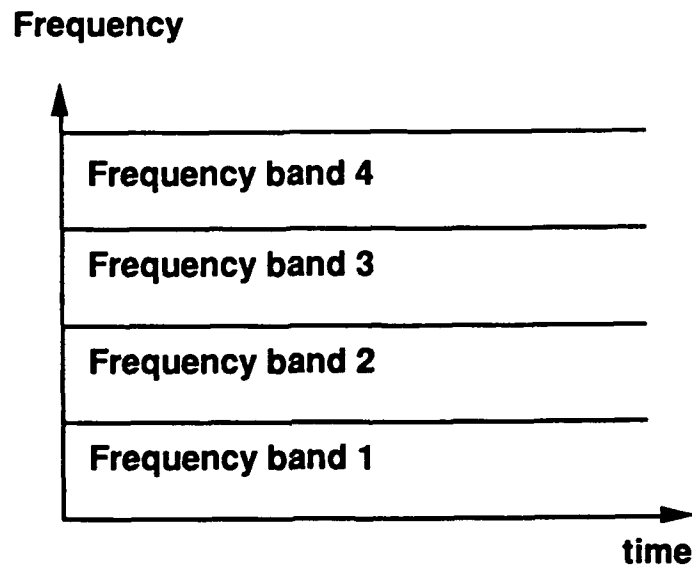


Fig. 2- Frequency Division Multiplexing

Packet switching is used to send data messages over the network. With packet switching there is no physical connection between two users. Instead a user sends the message into the network, knowing that it will reach its destination, but not know when or how it will get there. This is better suited for data information because it is usually delay insensitive and has a short hold time. For instance, if you want to send a chart showing last year's business profits to your boss, the information will not change if it takes a while to get to him through the network. Data information needs a higher bandwidth than voice information. Time Division Multiplexing (TDM) is used with data communication. TDM gives each user the whole frequency of the network for a limited amount of time (see Fig. 3).

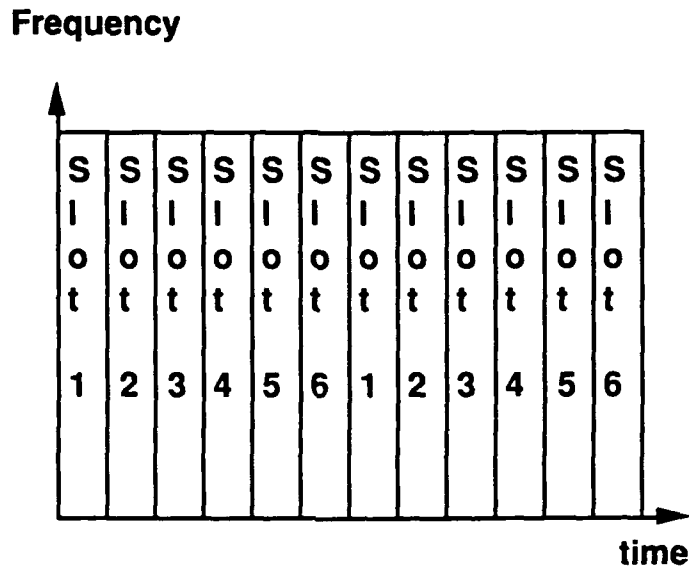


Fig. 3- Time Division Multiplexing

## 2.2 ISDN Applications

Many utilities can be linked to the ISDN. Local Area Networks (LAN's) can be bridged together. Computers can be used for Electronic mail between terminals, and for obtaining information from larger computers (e.g., mainframes residing at remote locations). ISDN will be better suited for this purpose because you will be able to access any resource hooked to the network as long as you have proper access privileges. Today different computers are linked to different networks and it is sometimes difficult to gain access to another network. Digital telephones can be used in the ISDN for end-to-end digital voice connectivity and users can hook special "ISDN" major appliances to the network, which may be turned on or off and controlled remotely. A home owner can have an alarm system on the house that can be monitored while away.

Figure 4 shows some of the services that will be available with ISDN. The integration of the utilities in the residential model will be very helpful. For example, someone can talk to a travel agent (the business model) over the telephone while looking at travel locations on their TV. The travel agent can then send airline prices to the user's PC and the user can evaluate the prices and purchase a ticket. All of this can be done from the home!

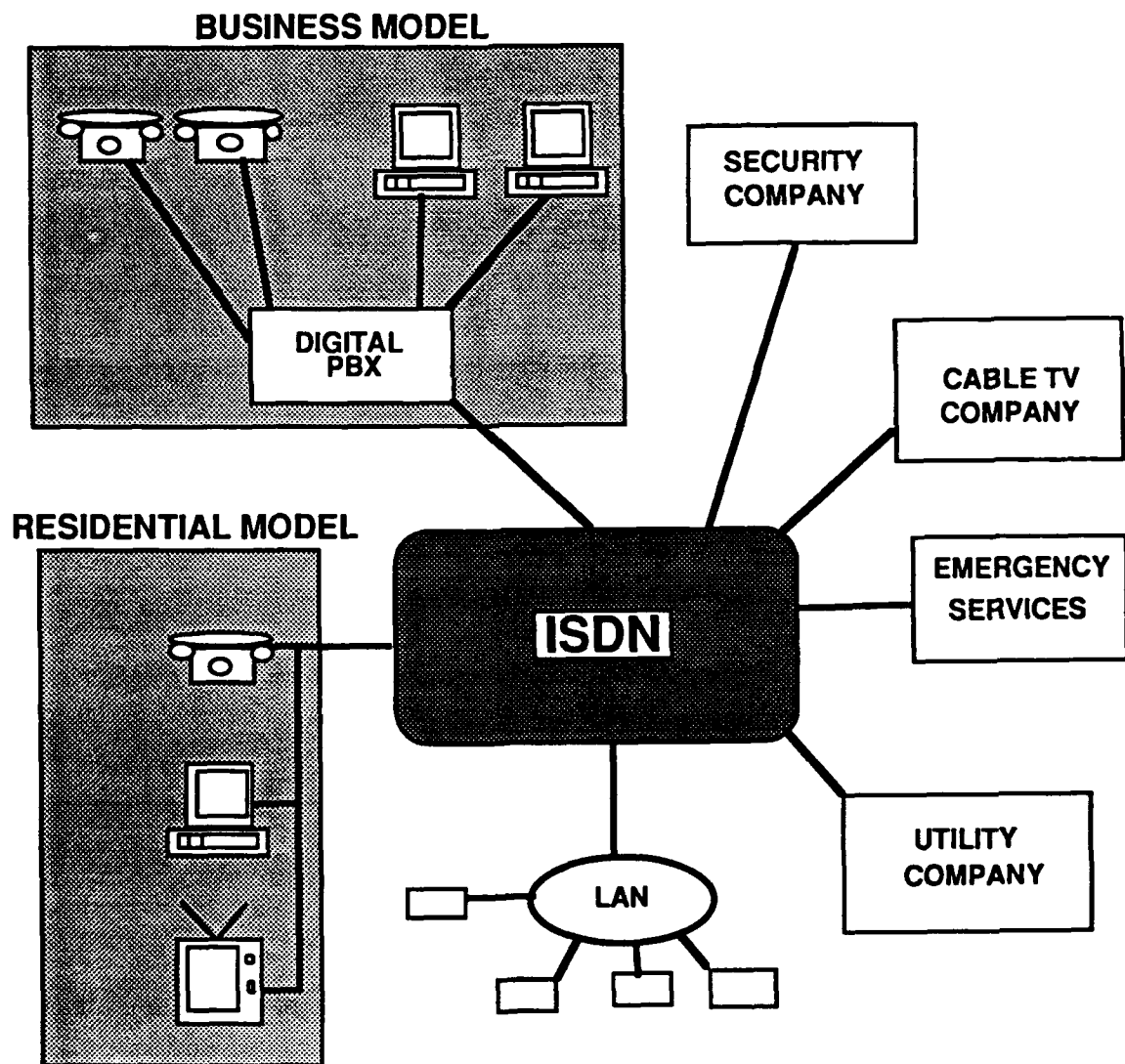


Fig. 4-ISDN services

Many people will benefit from the ISDN. Most importantly, it will be available to anyone who has a need for it, and can afford it. People can use it for better, faster, and more reliable communications. They can get better security systems that can be checked remotely and monitored by the police. Users will also have more services available to them, such as computer software, "mail order" services (done through the ISDN), catalog updates, etc. They can send billing information directly to the utility company and the company can check usage over the network. Businesses can use ISDN for distributing software, video (cable television), and communication services and for communication (eg. between store chains). They can also tell customers of sales, and show catalogs and updated prices over the network.

### 3.0 ISDN and the DoD

The military is interested in ISDN for communication purposes. They are mainly interested in the broadband aspects of ISDN because of the quantity of communication taking place between their widespread bases, posts, and command centers. ISDN will make military communication more reliable because military installations are spread worldwide and ISDN will become a worldwide network. ISDN will have a major impact on C<sup>3</sup>I functions. It will expand and improve information services to the user, improve mission capability by aiding architectural goals of survivability and interoperability, and decrease system support costs.

The network extension of services to the ISDN user is established because of high speed end-to-end digital connectivity. Because it will be used for many different services and these services can be used simultaneously (if necessary) network management is simplified. Crisis handling management will be greatly improved. For example, someone using the network for low priority business can be alerted, over the D channel, of a high priority message and take immediate action. Also, ISDN will make transmission of information more secure because it will no longer need analog to digital conversions. Today, voice quality is low because of the larger number of analog to digital conversions and low transmission speeds used on the networks. ISDN will allow information to pass at high transmission speeds and will have no analog to digital conversions, except, possibly, at the user interface. This is achieved with the use of sixty-four kb/s lines over an all digital network. Conference capabilities will also be upgraded due to enhanced voice quality using digital telephones.

Survivability and interoperability of C<sup>3</sup>I will be improved with ISDN. The network will be viewed as a communication "utility", much like electric power is today. With standard equipment and interfaces, and high bandwidth digital pipes, ISDN will provide much flexibility. Because the implementation of ISDN will be an evolutionary one, connecting other networks to ISDN will be possible. This will allow survivability of communication resources, from the national command level, to the base level, to be satisfied with military or commercial networks. Interoperability will be achieved



with ISDN because of the standard 64 kbps and the layered protocols. Even though the United States and Europe have different primary access channels, the basic user access of 2B+D will allow interoperability.

Finally, ISDN will decrease the cost of providing all communications services. This comes about with the use of common facilities and standard equipment. An end-to-end digital network will provide single line access to users for their voice, data, video and other needs. No additional equipment will be needed for analog to digital conversions through the networks. The use of standard equipment will improve maintainability and could affect a decrease in procurement lead times.

The DoD must be prepared to take advantage of ISDN. Many of the Defense Communications Agency's (DCA) programs will be affected with implementation of ISDN. The Defense Switched Network (DSN) and the Defense Data Network (DDN) will both be improved with ISDN. Communications and electronics professionals in the DoD should keep up with ISDN developments to allow consideration of the military's use of this technology. Also, the DoD must participate in the development and approval of ISDN standards. They must help develop the ISDN standards so they can use the full capabilities network to its full potential.

#### 4.0 ISDN Future

There are arguments for and against ISDN. One argument for ISDN is that the number of terminals and Public Business Exchanges (PBX's) has tripled between 1985 and 1990. Not all of these

terminals will need to be connected to a network, but there will be enough to make ISDN necessary. Also, it is predicted that the migration from analog signals to digital signals in the workplace will be nearly completed within the next five years. An argument against ISDN is that in 1990 the bandwidth requirements for voice and data were ninety percent voice and ten percent data. Some people don't think it is necessary to rush into a fully digitized network. Even if ISDN generates more data traffic, it is predicted that there will not be a huge increase until 1995.

In the future ISDN will become a fast, reliable way to transport data. One way this quickness and reliability can come about is with the use of optical fibers. Fiber optic cables are made of fiberglass and use light to transmit data. They have a higher bandwidth than today's cables, are lower in volume and weight (making them much cheaper), and because they are glass, they are immune to electromagnetic noise.

## 5.0 Summary

ISDN is something that is very necessary and will definitely be fully implemented in the not so distant future. It will take time, and will not combine all networks immediately, but when it does come, it will make communications easier, faster, and more reliable. The "Utopia" that many people picture when they think of ISDN will never fully come, because technology is ever growing and ISDN will have to grow with technology. ISDN will solve many of today's problems and answer many of its users' needs.

## REFERENCES

Dicenet, G. Design and Prospects for the ISDN. Boston: Artech House, 1987.

Kessler, Gary, C. ISDN. New York: McGraw-Hill, Inc., 1990.

Sippl, Charles, J. data communications dictionary. New York: Van Norstrand Reinhold Company, 1976.

Stallings, William. Tutorial: Computer Communications: Architectures, Protocols, and Standards. Washington, D.C.: Computer Society Press of the IEEE, 1987.

Stallings, William. Tutorial Integrated Services Digital Networks (ISDN). Washington, D.C.: Computer Society Press, 1988.

THE DETECTION OF LASING IN A DIODE CAVITY

Mr. Andrew Gerrard

8/9/91

## THE DETECTION OF LASING IN A DIODE CAVITY

Mr. Andrew Gerrard

This paper reports the spectral measurements performed on an external cavity diode laser with a Princeton Instrument's Optical Spectrometric Multichannel Analyzer. The ultimate result of the measurements being the detection of "lasing" within the cavity. The Analyzer was run with the Princeton Instrument's MSMA software package. The package contained the MSMA main program, the FSMA data-processor, and the SSMA step motor. The preliminary problems encountered with the setup and initiation of the project were the use of the program(s) and equipment, and the actual search for lasing within the cavity. Although no true lasing was discovered at the time of the writing of this report, great steps were being taken to improve the likelihood of observing lasing in the cavity.

In the past five years, great headway has been made in the field of optics. Since the days of the ancient Greeks, man has had an uncontrollable urge to discover the true nature of light, sound, and energy. It seems like only yesterday the first laser was built after years and years of research and speculation. While today in the optics and lasers area, leaps and bounds are being made every second. For the future? Who

knows, the possibilities are mind-staggering. But for now, lets leave the future for the philosophers and stick to the present. The experiment was designed with the intent to detect lasing in an external cavity diode laser using an Optical Spectrometric Multichannel Analyzer (OSMA) detector. In describing the experiment, it will be easier to first start with the overall concept and setup, then move on to the more mundane aspects and problems of the venture. Finally, a statement of conclusions and projections of this project will be discussed.

The main purpose of the experiment was to create an external cavity diode laser that would lase. The theory follows the same theory of the laser, with the diode emitting light and mirrors creating feedback in the diode/cavity system. In order to operate correctly, the cavity must be aligned perfectly to achieve proper feedback or lasing will not occur.

The setup of the laser experiment is a basic linear cavity (FIG A). In the middle of the cavity rests the laser diode mounted on a thin metal strip. A wire to the diode provides current from the laser diode controller to stimulate the active material of the diode to emit light. Since the diode is open at both ends, light can exit the diode from each side. Located at each end of the cavity is a mirror, one of which is totally reflective (the "back" mirror), while the other is 80% reflective (the "front" mirror, where the lasing

beam would pass through). The mirrors feed the light back into the diode, therefore causing the light to oscillate back and forth in the cavity. The partially reflective mirror allows 20% of a beam to pass out of the cavity, which will then be detected by the OSMA detector.

However, because of the equipment layout and the need to be able to make adjustments, the beam first had to be routed over the optical bench with a series of mirrors in order to be incident on OSMA spectrometer and detector head. The mirror combination moved the beam over 51cm and raised it 5cm. After the beam was aligned, two irises were added for calibration and alignment purposes.

In order to use the OSMA, a MSMA software package from Princeton Instruments, Inc., was utilized. Instead of using the SSMA, we relied on the front panel of the spectrometer, which was easier than constantly switching programs. However, the use of the MSMA and FSMA programs were vital to the detection stage as the original intent to use a fiber optic coupled optical spectrum analyzer was unsuccessful, simply because its data update rate was too slow to detect lasing if it occurred. The MSMA programs allowed almost instantaneous update, thus improving the possibility of observing lasing.

However, the use of the OSMA and MSMA software caused certain problems. First, the OSMA detector is very sensitive, almost too sensitive for our purposes. More so, the detector becomes more sensitive as one moves the beam toward the upper

area of the spectrometer entrance slit, so that near the top, saturation of the detector is inevitable. Second, a slight change in the angle of the incoming beam also changed the measured spectrum. Putting these combinations together caused endless possibilities, making lasing even more elusive (the addition of the two irises helped some, though more time had to be spent re-aligning the beam if adjustments were made to the cavity).

The problems with the MSMA and FSMA programs were their confusing operation methods and "foggy" characteristics. The programs utilize many unintelligible commands and have a very user UNfriendly interface, and unnecessary time was spent trying to decipher the programs. However, after time consuming experimentation, using the programs becomes much more clear and second nature. Appendix A lists the common commands of the MSMS software.

Other problems that occurred with the cavity setup dealt mainly with alignment. Every change in the cavity (ie. adjustment of the mirrors or objectives, etc.) caused the beam to travel out of the irises and detector head. This, if allowed to continue, would wreak havoc upon the measured spectrum and hence cavity mode patterns. Constant changes once again had to be made with each step to the cavity alignment process in order to maintain optimal alignment with the OSMA detector.

In order to minimize the variables of the experiment, the



setup was kept as constant as possible. The current going into the diode was kept at 292.0mA. The resulting beam was attenuated with two optical density filters (OD: 1. and 3. respectively) so that the detector would not saturate. Irises and the focusing lens going into OSMA were also kept constant. As with any experiment, as few factors were changed at a time as possible, reducing measurement error.

At the time of the writing of this paper, no "true" lasing had yet been found. Main results obtained were the discovery of small mode patterns (FIG B) on the wave. Though this was the start of lasing, nothing more could be obtained. Later, it appeared that the laser diode did lase on its own, however the result was not the intended one. Causes for the lose of lasing are seemingly small in number. Alignment was optimal and the beam was always aligned with the irises and detector head. The beam was kept at the constant power, and conditions within the lab were kept at high standards. However, there could have been a problem with the setup of the laser diode or the cavity itself, perhaps even the OSMA detector was at fault. In any case, the lasing we were looking for did not appear, though even as I write, steps are being made to "correct" the problem.

In conclusion, the project was neither a success nor a failure. Although any number of small errors could disrupt the lasing, I believe few were present in cavity, diode, or mirror combinations. Great insight was gained from the experiment ,

and it is only a short matter of time before lasing is discovered within the cavity.

# EXTERNAL CAVITY DIODE LASER

FIG. A

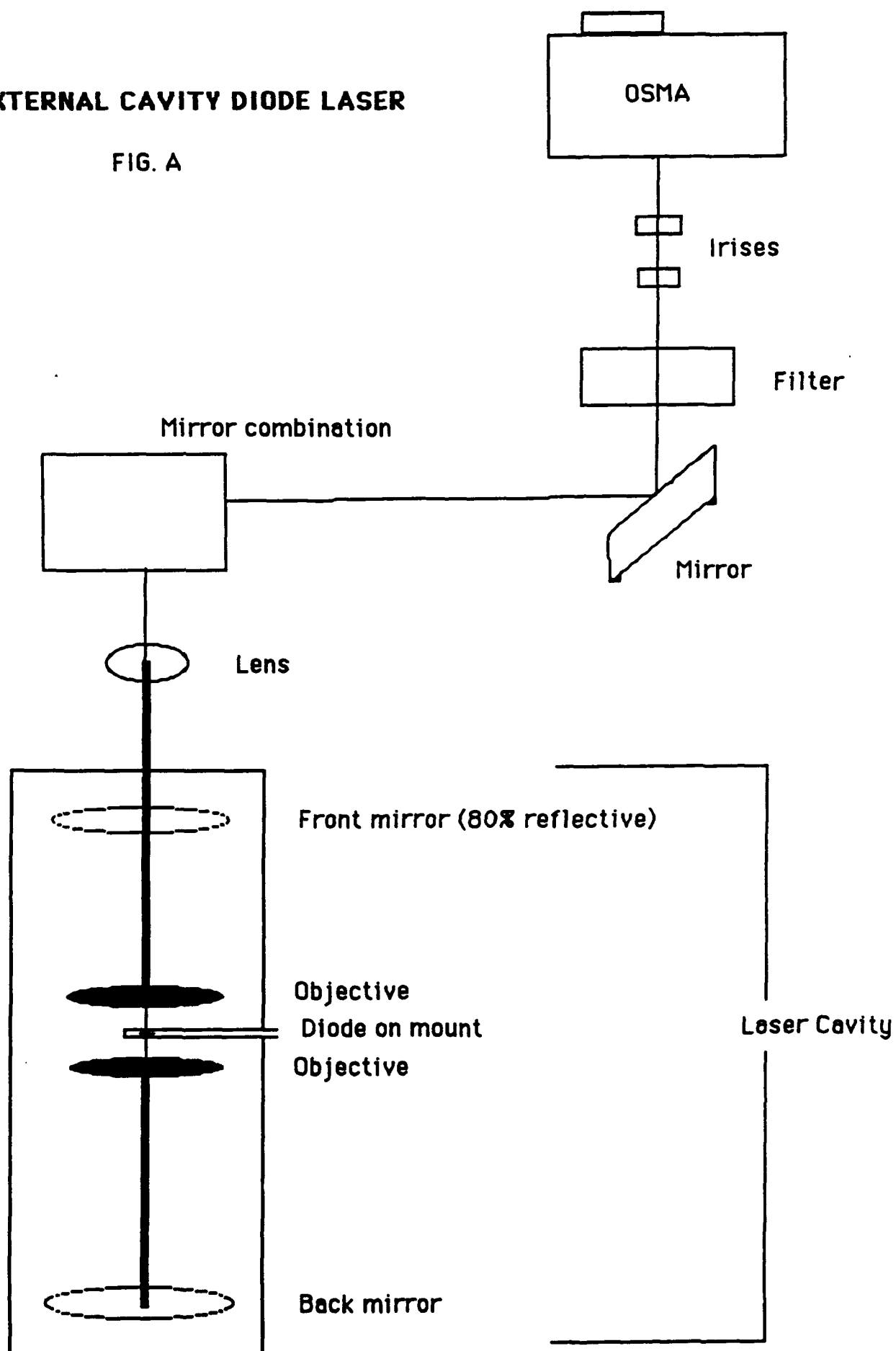
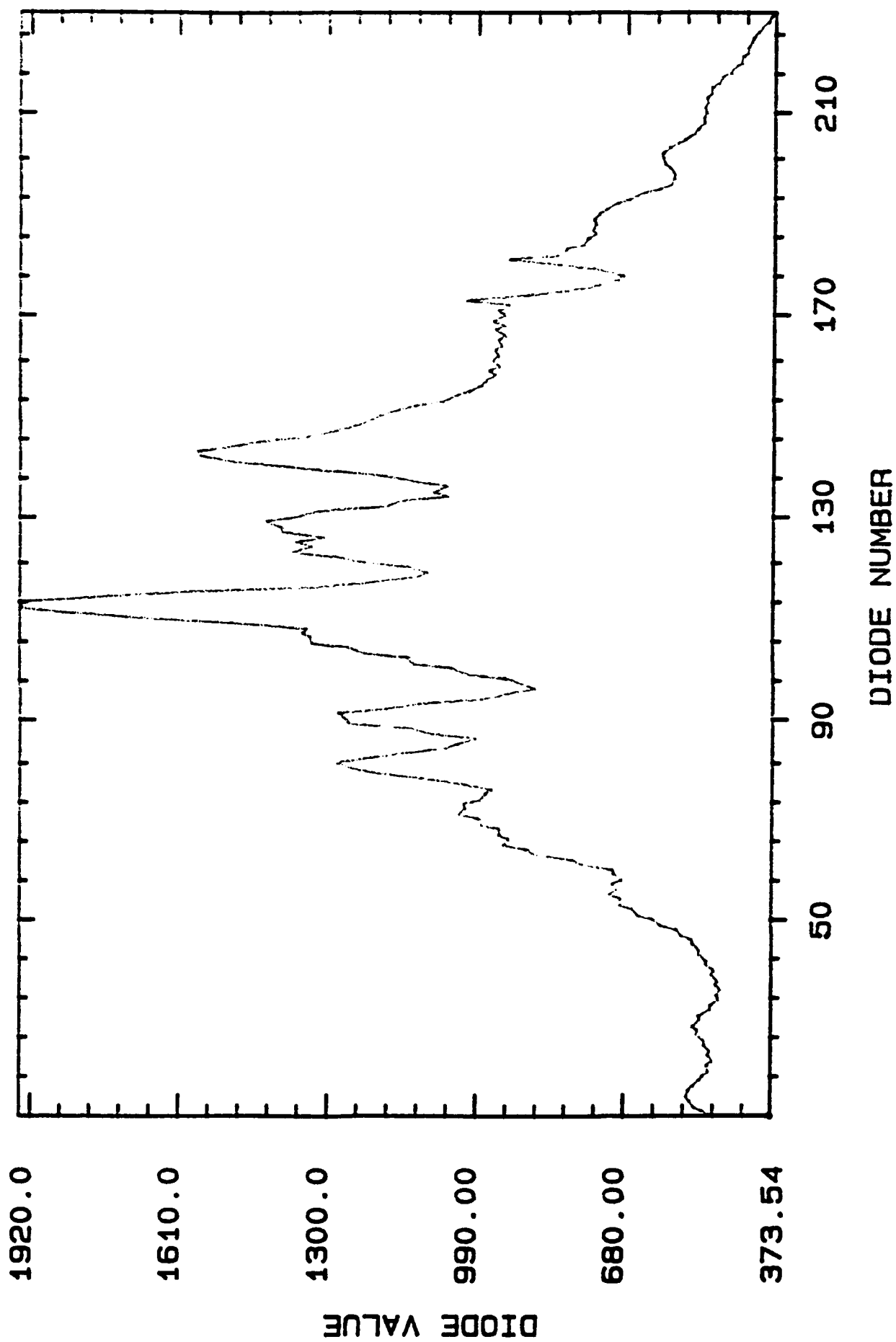


FIG. B



## APPENDIX A

### Common Commands of MSMA

Commands	Meanings
Experiment-New	Shows a current experiment setup
Experiment-saVe	Saves current setup creates .XST file
Experiment-Load	Loads a setup
Experiment-Run	Runs experiment under current setup conditions
Experiment-Stop	Stops experiment, but does not store data
Display-File_replay -File_replay	Replays experiment's stored data
Display-File_replay -stacked	Allows multiple spectra display (simply change file name)
Display-File_replay -Spec	Use before making calibration, used for multiple spectra
Display-Line_mode	Toggles dots/line patterns
Display-set y-max/min	Sets Y-scale
Window-Erase	Erases current window
Window-Open/Close	Opens/Closes windows
Window-Reset -All_windows	Resets all windows to norm.
Window-Reset -Stop_replay	Stops file replays (can be used with Erase)
pRocess-cAlculator	Sets up spectrum calculator

into	pRocess-conVert	Converts data from MSMA another form !!! convert into float for data processing!!!
	pLot-plot	Use serial port 1 "what you see is what you get!"
	pLot-setup	Standard plot setup area
	Special-data_Process	Goes to FSMA
	Special-step motor	Goes to SSMA
	Special-Configure	Sets system configuration

#### Commonly Used Function Calls in MSMA

---

F2	Toggles Y-axis
F3	Horizontal zoom in
F4	Horizontal zoom out
F9	STORES DATA!!! and ends experiment creates .SPE file

## Commonly Used Procedures in MSMA

-----

### To run an experiment

- 1) Special-Configure  
Make sure sysyem is set
- 2) Window-Reset-All windows  
Resets window to normal conditions
- 3) Experiment-New  
Set experiment setup as needed
- 4) Experiment-saVe  
Saves ONLY setup, not data
- 5) Experiment-Run  
Runs experiment as to setup
- 6) F9 to store data  
or  
Experiment-Stop to not store data

### To calibrate experiment

- 1) Make sure you have already run an experiment and have STORED (F9) it
- 2) Display-File\_replay  
-Spec\_display
- 3) Go to Calibrate menu
- 4) Go to graph tracking mode on mouse
- 5) Go to first area and hit "1"
- 6) Enter value for area and hit enter
- 7) Go to next area and repeat
- 8) Save into calibration file on same menu
- 9) Go to Experiment-New
- 10) Add the name of calibration file

### To subtract background

- 1) Make sure you have already run a background experiment and have STORED (F9) it
- 2) Go to Experiment-New
- 3) Enter background file name and switch to "Yes" on setup menu

- TO convert data
- 4) Run experiment
  - 1) Make sure you have already run and STORED an experiment
  - 2) pRocess-conVert-Float
  - 3) Enter new filename

- To use calculator
- 1) Use as normal calculator
  - 2) Use Window-Open and File replays as needed BEFORE usage
  - 3) Enter a "0" for all spectra

example:

[ test1,1,1 - test2,1,1 = ,1,1 ]

meaning:

file test1, 1 spectra - file test2, 1 spectra  
to be displayed(no file name given) onto  
the screen

- 4) Save equations as needed
- 5) Hit " = " to process equation, with "Enter" to display onto screen or a file name to store answer in file

- To plot
- 1) MSMA operates on a " what you see is what you get " basis
  - 2) pLot-Setup as needed
  - 3) pLot-Plot and again set screen as needed
  - 4) Plot command

Note: plotter must be in serial port 1

There is a Command Glossary, Key Chart, etc...  
on page 3-1 in the software section of the  
OSMA manual. Note however that this glossary  
is for use on OSMA and some commands are  
have been changed in the newer MSMA.



## Common Commands of FSMA

---

### 1) Graphics and Plots

- Make sure that data in file is stored and converted into Floating values before graphing
- When marking a position on a graph, be sure that you use the "END" key instead of the "Enter" key
- Simply follow the commands that are given by FSMA
- other graph commands:
  - Zoomin: allows user to view a specific area
  - Area: allows user to obtain peak areas, peak heights, and ratios of these values
  - Original: returns to the original graph
  - Next Experiment: allows plotting of another experiment without having to go back to the main menu
  - Plot: plots graph (make sure system configuration is set at desired setting)  
Enter headers as needed (along with comments), or hit "Enter" to bypass
  - End: returns to main menu

### 2) Data Processing

- Make sure all data is in Floating point values
- Follow commands as needed

### 3) Create, Modify, or Run User Programs

- Use as needed

### 4) Edit Experiment Information and Data

- Allows user to change collected data

### 5) Recall Experiment Information and Data

- The user can save data into a text file, must be float
- The user can recall data collected

### 6) Modify System Configuration

- Sets system for operation within FSMA only

# C-ing The SUN



**Rome Labs/ERDA  
Griffiss Air Force Base  
Research & Development Laboratories  
Mentor: Paul Ratazzi**

*by Todd Gleason  
August 15, 1991*

## Abstract

This summer I worked not on the Digital Analysis System (DAS) testing A/D converters but instead wrote C programs to automate various functions. As I had no knowledge whatsoever of C, most of my time was spent learning it and writing very simple programs in C. A great deal of debugging was required, as C has a very strict syntax which is not easy for novices to work with. I wrote a fair game program to generate a pseudo-random number and let the player guess it. The second (and perhaps the one I liked the most for its simplicity and possible use) was a sort of screen saver which leaves a message on the terminal when run. The third and final program, which I was unable to finish, was a menu interface complete with windows which allows the user to receive files from the DAS using Kermit as well as letting the user convert and merge those files to a format whereby FFT's (Fast Fourier Transforms) can be run on them. I used our Sun workstation to write the C programs, with the Open Windows and SunView programs. In addition, I interfaced with the Sun workstation via the LONEX terminals, and edited files from there using the VI program.

## Introduction

In the course of my eight-week apprenticeship, I have been able to greatly expand my knowledge in relation to a few fields of work. I learned how to use the Sun workstation and ~~about~~ the UNIX system, and found it to be somewhat similar to MS-DOS, which I was already quite familiar with. I also logged into the Sun workstation through LONEX, a computer network, where I learned to use the VI editor to write my C programs. I used the curses.h library for creating and manipulating windows, and the string.h library for manipulating the character strings for such things as messages and file name i/o handling. By running my programs (and subsequently dumping the core many times), I was able to find out what I could do to improve their flow and fix their errors. Although I found C's low-level processor a hassle to work with and was rather dismayed at its method of handling strings and pointers, I found C to be an efficient language, and am glad I had the curses.h library to work with.

# Discussion

## Learning C Language

The task I worked on this year was learning C and applying my knowledge to write programs for the Sun computer. As I had no prior knowledge, I had to begin from scratch. It was necessary to use C language because the Sun uses a C-type command processor; i.e., it operates in a manner which is compatible with C and many of its capabilities and syntax are similar to that of C.

Fortunately, we had a few resources for me to study. Dave, one of the engineers, had an introduction to C in a 3-ring binder of photocopied pages from a lecture, so I borrowed it and began reading through it. Many of the functions initially confused me, such as the format for the `scanf()` function. I had trouble understanding why the pointers such as `&` in front of `var1` needed to be used.

By far, the largest problem I had was with defining functions. The format was in no book I found (they all used somewhat different formats) and I had to ask my mentor. However, it was very annoying, as I often had to wait a while. (My mentor was away at conferences for about three weeks of my time here.) This forced me to rely more upon myself, which helped me a great deal in structuring my programs. However, it also hurt me in that when I had difficult questions there was no one who knew the answers for certain.

The first program I wrote was a simple game to generate a random number from 1 to 1000 and ask for a guess. It would inform the user that the number was higher or lower until the input was correct. Unfortunately, there was much difficulty in getting the program to generate truly random numbers. On an IBM machine in BASIC this is a matter of `RANDOMIZE TIMER`; on a UNIX machine in C I had to try to call functions, and was not able to generate very random numbers. The program worked, and subsequently excessive error-handling was put into it, taking much more time to code in.

After the game, I knew I still was not ready for a menuing program, so I started writing a screen saver program, something that seemed useful as well as being a small chance for me to try using the `curses.h` package. I knew that writing something to actually clear the screen and restore it would not be a simple task, so I wrote something that was—in my opinion—more useful, although it still was a sort of screen saver. The program initially cleared the screen, drew a box around the border, printed a message in the center of the screen, and drew a box within the border's box. The box was animated; the program drew the horizontal lines with "-" and the vertical lines with "|" and when it got back to the upper left of the box, it wrote over them with the reverse. This resulted in a simple effect which also kept any screen from blanking. This way, anyone who approached the computer would see it, even if it was run from within a window in SunView or OpenWindows, as the display would attract attention. Then they would be informed that the

that the user intended to return to the computer momentarily.

Later versions changed such things as the animation set to make it loop through four sets of characters to use for the border and actually read the user login id from the Sun. This made the program more generic--anyone could use it, and it would simply print their login id below the message. Screen garbage was also cleared up. The program could be upgraded to input a custom message each time and check for a simple carriage return to optionally print a default message.

The final program was extremely difficult for me to write, having had only weeks of C language experience. The proposed function of this program was to allow the user to more easily receive files from the DAS 9200 computer using the Kermit program and protocol, convert and merge the files together, and perform FFT's all from one window menu interface.

Setting up functions was the most time-consuming part of my work on this program, next to debugging. I was forced to go back through the entire program and change all the sets of variable introductions and declarations each time I added, changed, or removed a global variable.

Eventually, I got all the windows in the right places and refreshed them (clearing and redrawing) at just the right places so that they didn't need to be refreshed too many times (which considerably slows down the program interface on a remote terminal). I wrote the inputting function for file names and the Kermit file transfer shell selection. In addition, I attempted the shell for conversion and merging of files, but

had difficult problems there. I was unable to make the program automatically select an output file. For instance, Dave used filenames like in1-095 and in2-095 which needed to be converted and merged into in-095. To do this, I needed to look for the dash and copy all characters *except* the character before that dash. I was unable to do this, despite numerous attempts at using the strncpy command, due in part to lack of time and in part to lack of assistance—even my mentor was unsure what might work in the complicated subroutine I had to write. I finally got it to where the program seemed to know the output file, but would still not work correctly; it simply returned to the menu without doing anything. Still, the program seems fair based on the time at hand.

## **Understanding the Sun**

Learning how to use the Sun itself was also a necessary step, and a time-consuming one at that. I began without an account on my first day, as my mentor wasn't there. As a result, I paged through some of the guides to using the Sun. However, without a machine to work on, it was sometimes difficult understanding what the books were talking about. Other times the books were easy to understand either because they were discussing aspects of the SunOS that were similar in operation to that of IBM machines or were simply easy to understand rules, sometimes with excessive explanations.

The Sun has a command line interface similar to that of MS-DOS.



When I logged on and gave my password, it ran my .login file and my .cshrc definitions, printing such things as a listing of who else was logged onto the Sun (which is called the *finger* command) and then printed out a fortune from a library almost 350K. After that, it came to the "bugs%" prompt, which I later changed to read, "bugs%51>", where the 51 was the number of my command. (The Sun keeps a *history* of user-definable length, and previous commands can be easily recalled using !, !vi, or !50, for instance. !! recalls the last command, !vi recalls the last command with the string "vi", and !50 recalls command 50. Upon each logoff, the sequence is saved and the last 50 commands are renumbered 1-50 for the next login.

The *vi* editor was perhaps the most difficult aspect of the Sun to get used to, aside from the C programming. The reason for this is its strange interface, designed to be compatible with the old vt100 and vt52 terminals we have at work. To illustrate, the program begins in *command* mode. Pressing "I" activates *insert* mode. "A" appends text. This allows text to be added onto the end of lines. The "ESC" key exits these, allowing you to move around. From *command* mode, pressing ":" or "/" activates *last-line* mode. The ":" allows using the "w" to write a file, the "r" to read and insert a file, the "q" or "q!" to quit, and "x" to exit. There are many other options but these are the most used. The "/" allows search and replace functions. As this differs from most editors, which allow the user to cursor around and simply type text into any space, it is difficult to use. However, it is very compatible with all

terminals and can even do certain actions (such as "dd" to delete a paragraph) on older terminals by using an "@" as a place-holder for a blank line, and by not moving text that you are inserting before but waiting until *insert* mode is exited to do that.

Using SunView was a fairly easy to get used to process. I could simply move the mouse from one window to another and the cursor would appear there for me to do my typing conveniently. If I began one procedure I could leave it going and work on something else. Or I could have multiple windows open to allow me to simply have more than one application up at a time.

OpenWindows was very similar to SunView in its usage, although it had a much more flashy interface complete with 3D frames and buttons on its windows. I used OpenWindows more often as my work progressed, and using it I would often save a C program from the Text Editor, move to the Command Tool and compile it, and finally move to the Shell Tool to run my program. In the latter two windows, I could often just type "!!" to repeat commands, further saving time.

Later on, I used the graphical utilities of the OpenWindows program to help me with getting the Sun logo on the title page. I manually drew in the dots, doubled it in size and smoothed it back out.

## Results

On the following pages are the last two programs, as they were at the writing of this Report, complete with comments for those not familiar with C programming. Although someone foreign to C will not be able to fully understand, the comments should aid in making a general analysis as to the nature and flow of the programs. It was impossible to include the random number guessing game do to lack of space. A sample screen from the menu program *is* included, however.

```
#include <curses.h>
#include <sys/types.h>
/* Screen Saver Program--Identifies that you are using terminal while
you are away so that no one logs you off thinking that you're not
coming back. Creates a cute display of a box around the screen with
a message (*msg which can be replaced with user's name) centered.
Then creates another box, slightly closer in, which is animated by
a changing pattern of symbols used as box characters. */
main()
(
    int vert, hor, loop, a, b, c, d, e, f, g, h; /* Starts declaring integer
    variables */
    int row,col;
    char *msg = "I will return shortly."; /* Default Message to be placed in
    center of screen */
    int nul = 32; /* Space character to use */
    initscr(); /* Begins use of curses.h library */
    vert = 124; /* Sets vertical component of box to | */
    hor = 45; /* Sets horizontal component of box to - */
    a = 124;
    b = 45;
    c = 92;
    d = 47;
    e = 43;
    f = 88;
```

```

g = 33;
h = 42; /* a-h are sets for the inner box to use as border characters */
box(stdscr, vert, hor); /* Draws main screen box */
refresh(); /* Updates screen */
mvaddstr((LINES/2 - 1), (COLS/2 - 1 - (strlen(msg)/2)),msg); /* Prints
    default message */
mvaddstr((LINES/2 + 2), (COLS/2 - 1), cuserid("")); /* Prints user login
    id name */
while (1) /* Loops forever until Break is signalled */
{
    for(loop = 1; loop < 5; loop++) /* Sets loop variable for various character
        border sets */
    {
        if (loop == 1)
        {
            vert = a;
            hor = b;
        }
        if (loop == 2)
        {
            vert = c;
            hor = d;
        }
        if (loop == 3)
        {
            vert = e;
            hor = f;
        }
        if (loop == 4)
        {
            vert = g;
            hor = h;
        }
        for((row = 2), (col = 3); col < (COLS - 4); col++) /* Top side of box */
        {
            mvaddch(row, col, hor);
            refresh();
        }
        for((row = 2), (col = (COLS - 4)); row < (LINES - 3); row++) /* Right side
            of box */
        {
            mvaddch(row, col, vert);
            refresh();
        }
        for(row = (LINES - 3), (col = (COLS - 4)); col > 2; col--) /* Bottom side of
            box */

```



```

msg = newwin(7, 42, 16, 35);
minput(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn); /* Moves off to minput function */
}
int drawscr(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn)
    int ver, hor, i, ierror, nul, respn;
    char *respc, *achoice[ACHOICES][35], *aprompt[25], *file1, *file2,
        *fprompt, *msghdr, *msg1;
    WINDOW *cinput, *finput, *msg, *stdscr; /* Declares function drawscr
        and variables to import from calling function */
{
    /* Draws the Screen after an operation is finished. */
    clear();
    box(stdscr, ver, hor);
    for(i = 0; i < ACHOICES; i++)
    {
        mvwaddstr(stdscr, (2 * (i + 1)), 3, achoice[i]); /* Prints menu choices
            after moving to different positions on main window */
    }
    mvwaddstr(msg, 1, 3, msghdr); /* Prints header for msg window */
    if (ierror == 0)
        strncpy(msg1, "READY", 20); /* Default message */
    if (ierror == 1)
        strncpy(msg1, "Value out of range! Try again!", 20); /* Error message */
    if (ierror == 2)
        strncpy(msg1, "Enter Filename 1:", 20); /* Prompt message */
    if (ierror == 3)
        strncpy(msg1, "Enter Filename 2:", 20); /* Prompt message */
    mvwaddstr(msg, 3, 3, msg1);
    wclrtoeol(msg);
    box(msg, ver, hor);
    mvwaddstr(finput, 1, ((20/2) - ((strlen(fprompt))/2)), fprompt); /* Centers
        header for finput window */
    mvwaddstr(finput, 3, 5, file1); /* Prints first file, indented */
    wclrtoeol(finput); /* Clears to end of line in case of extra text */
    mvwaddstr(finput, 4, 5, file2); /* Prints second file, indented */
    wclrtoeol(finput); /* Clears to end of line in case of extra text */
    box(finput, ver, hor); /* Prints finput window box */
    mvwaddstr(cinput, 2, 3, aprompt); /* Prints main input prompt */
    box(cinput, ver, hor); /* Prints box around cinput window */
    wrefresh(stdscr); /* Updates main screen window */
    wrefresh(msg); /* Updates msg window */
    wrefresh(finput); /* Updates finput window */
}

```

```

wrefresh(cinput); /* Updates cinput window */
return; /* Returns to calling function */
}

int minput(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn)
    int ver, hor, i, ierror, stdscr, nul, respn;
    char *respc, *achoice[ACHOICES][35], *aprompt[25], *file1, *file2,
        *fprompt, *msghdr, *msg1;
    WINDOW *cinput, *finput, *msg; /* Declares function minput
        and variables to import from calling function */

{
    while (1) /* Loops forever */
    {
        drawscr(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn); /* Draws the screen */
        scanw("%d", &respn); /* Scans for numeric input */
        if (respn == 1)
        {
            choice1(ver, hor, i, ierror, stdscr, nul, file1, file2,
respc, aprompt, achoice, cinput, finput, msg, fprompt, msghdr,
msg1, respn); /* Calls function choice1 */
        }
        else if (respn == 2)
        {
            choice2(ver, hor, i, ierror, stdscr, nul, file1, file2,
respc, aprompt, achoice, cinput, finput, msg, fprompt, msghdr,
msg1, respn); /* Calls function choice2 */
        }
        else if (respn == 3)
        {
            choice3(ver, hor, i, ierror, stdscr, nul, file1, file2,
respc, aprompt, achoice, cinput, finput, msg, fprompt, msghdr,
msg1, respn); /* Calls function choice3 */
        }
        else if (respn == 4)
        {
            choice4(ver, hor, i, ierror, stdscr, nul, file1, file2,
respc, aprompt, achoice, cinput, finput, msg, fprompt, msghdr,
msg1, respn); /* Calls function choice4 */
        }
        else if (respn == 5)
        {

```

```

        choice5(vert, hor, i, ierror, stdscr, nul, file1, file2,
        respc, aprompt, achoice, cinput, finput, msg, fprompt, msghdr,
        msg1, respn); /* Calls function choice5 */
    }
    else if (respn == 6)
    {
        choice6(vert, hor, i, ierror, stdscr, nul, file1, file2,
        respc, aprompt, achoice, cinput, finput, msg, fprompt, msghdr,
        msg1, respn); /* Calls function choice6 */
    }
    else if (respn < 1 || respn > ACHOICES) /* If there is an error */
    {
        ierror = 1; /* Sets error message */
        drawschr(vert, hor, i, ierror, stdscr, nul, file1, file2,
        respc, aprompt, achoice, cinput, finput, msg, fprompt, msghdr,
        msg1, respn); /* Draws the screen */
    }
}

int choice1(vert, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn)
{
    int vert, hor, i, ierror, stdscr, nul, respn;
    char *respc, *achoice[ACHOICES][35], *aprompt[25], *file1, *file2,
    *fprompt, *msghdr, *msg1;
    WINDOW *cinput, *finput, *msg; /* Declares function choice1
    and variables to import from calling function */

    if (ierror == 1)
    {
        clear(); /* Clears the screen--more thorough than erase function */
    }
    else
    {
        wclear(cinput); /* Clears the cinput window */
        erase(); /* Erases the screen */
    }
    strncpy(file1, "", 15); /* Erases the file1 variable */
    strncpy(file2, "", 15); /* Erases the file2 variable */
    ierror = 2; /* Sets input message */
    drawschr(vert, hor, i, ierror, stdscr, nul, file1, file2, respc,
    aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
    respn); /* Draws the screen */
    /* Now get files for input */
    mvwaddstr(finput, 3, 3, "");
    wrefresh(finput); /* Refreshes finput window */
}

```



```

scanw("%s", file1); /* Scans for first filename */
ierror = 3; /* Sets input message */
drawscr(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn); /* Draws the screen */
mvwaddstr(finput, 4, 3, "");
wrefresh(finput); /* Refreshes finput window */
scanw("%s", file2); /* Scans for second filename */
ierror = 0; /* Sets main input message */
/*drawscr(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn);
refresh();*/
respn = 0;
return; /* Returns to calling function
}

int choice2(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn)
    int ver, hor, i, ierror, stdscr, nul, respn;
    char *respc, *achoice[ACHOICES][35], *aprompt[25], *file1, *file2,
        *fprompt, *msghdr, *msg1;
    WINDOW *cinput, *finput, *msg; /* Declares function choice2
        and variables to import from calling function */
{
    clear(); /* Clears the screen */
    mvwaddstr(stdscr, 1, 1, "Receiving files via Kermit...\n\n");
    refresh();
    system("cd /home/bugs/dave/marf \n r_refmem"); /* Shells out to r_refmem
        program */
    return; /* Returns to calling function */
}

int choice3(ver, hor, i, ierror, stdscr, nul, file1, file2, respc,
aprompt, achoice, cinput, finput, msg, fprompt, msghdr, msg1,
respn)
    int ver, hor, i, ierror, stdscr, nul, respn;
    char *respc, *achoice[ACHOICES][35], *aprompt[25], *file1, *file2,
        *fprompt, *msghdr, *msg1;
    WINDOW *cinput, *finput, *msg; /* Declares function choice3
        and variables to import from calling function */
{
    char *listset[75], *file3[15]; /* Declares additional variables */
    int n, dash;
    clear();
    mvwaddstr(stdscr, 1, 1, "Converting/Merging files...\n\n");
    refresh();

```

```

for (n=1; n<=(strlen(&file1)); n++)
{
    if (&file1[n] != "-") /* Searches for a dash */
    {
        /*file3[(n-1)] = &file1[(n-1)];*/
        mvwaddstr(stdscr, 5, 3, "Made it to loop 1!"); /* Notification message
        for debugging purposes */
        refresh();
        if (dash != 1)
        {
            mvwaddstr(stdscr, 7, 3, "Made it to loop 2! "); /* Notification
            message for debugging purposes */
           printw(stdscr, "File3 = %s\n", file3);
            refresh();
            strncat(file3, &file1[n-1], 1); /* was file3[n-1] */
        }
        else strncat(file3, &file1[n-1], 1); /* was file3[n-2] */
    }
    else dash = 1;
}
/* strcat(file3[(strlen(&file1)-2)], &file1[(strlen(&file1)-1)]); */
/*file3[(strlen(&file1)-2)] = &file1[(strlen(&file1)-1)];*/
strcpy(listset, "cd /home/bugs/dave/marf \n x_refmem "); /* Copies to string
for shell execution */
strcat(listset, file1); /* Appends string for shell execution */
strcat(listset, " ");
strcat(listset, file2);
strcat(listset, " ");
strcat(listset, file3);
system(listset); /* runs shell */
refresh();
return; /* Returns to calling function */
}

```

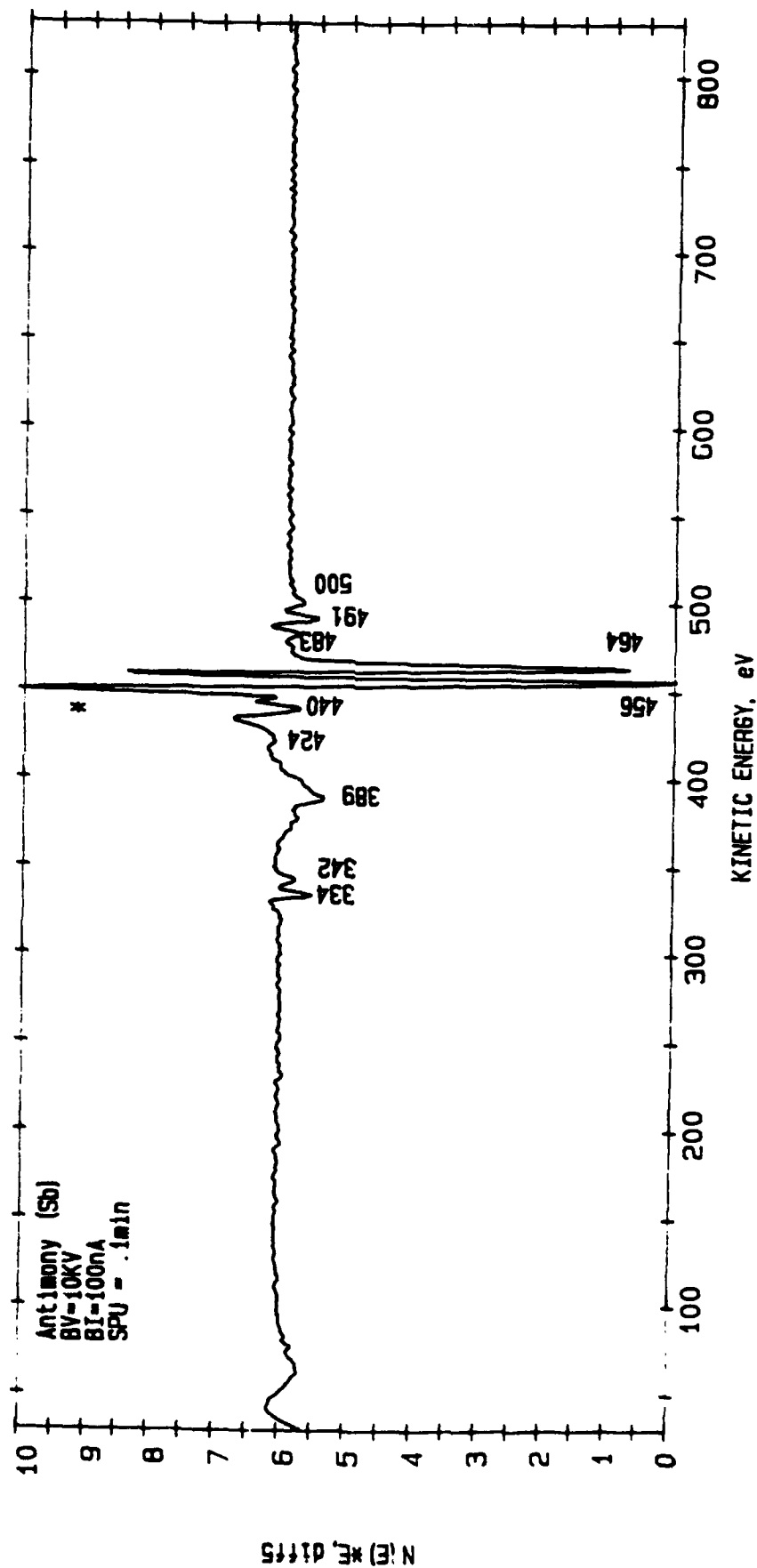
(Dummy functions deleted to conserve space.)

Figure 3

AES SURVEY P-C 6/28/91 AREA 1

FILE: sb10.100d Original Filename: sb10.100d Massage

SCALE FACTOR= 5043.824 k c/s. OFFSET= 0.000 k c/s



## **Conclusion**

I must admit that the beginning of this apprenticeship did not look too promising. However, as time progressed, and I learned more C and more about the Sun I grew much more accustomed to both and now I believe that this summer was much better than the last. I learned more, had some fun doing so, and additionally the C knowledge will be helpful in the future as references for college and for jobs. My knowledge of file transfers between systems may also augment my capabilities in college.

I would lastly like to thank all the wonderful people who made this possible for me. Their names are:

Paul Ratazzi (my mentor, who helped debug so many trivial errors)

Dave Williamson (a college graduate who works with the A/D chip and the MARF module and with various simulation and analysis software on the Sun workstation)

Mike Kopack (a high school graduate student working in a fellowship program who worked on interfacing the LTX with a new 386-25 system with a PCXNET program)

Nancy Koziarz (another of my superiors, who worked more with Dave Williamson and Paul Ratazzi than with me)

Julie Brichacek (another of my superiors, who let me borrow her C manual and helped me with the LONEX system)

## **Resources**

### **Used During Job**

1. NTU Advanced Technology & Management Programs, *C Programming for Engineers: Introduction*. Lecture by Dr. F.S. Hill, Jr, 1991.
2. Sun Microsystems, Inc., *SunOs: An Introduction*. Sun Microsystems, 1989.
3. Sun Microsystems, Inc., *SunOs User's Guide: Getting Started*. Sun Microsystems, 1990.
4. Sun Microsystems, Inc., *Programmer's Overview Utilities & Libraries*. Sun Microsystems, 1990.
5. Sun Microsystems, Inc., *Programmer's Language Guides*. Sun Microsystems, 1989.

AUGER STANDARDS FOR THE  
SCANNING AUGER MULTIPROBE PHI 600

by

Julie A. Gopsill

## Abstract

This document presents standards for certain elements and compounds commonly found in microelectronics as well as basic properties of all samples to help with the identification and quantification of elements and compounds with the Scanning Auger Multiprobe (PHI 600).

These standards were acquired using the Scanning Auger Multiprobe (PHI 600) and samples of known composition (Standards for Auger Electron Spectroscopy, registered standard no. 2128). The Auger electron is a result of an electron beam interacting with a sample. Each element (except hydrogen and helium) has a unique Auger electron pattern and, therefore, can be identified by matching the unknown spectra with the spectra of a known element. Some of the spectra characteristics are determined by the analyzer used in the acquisition process. Therefore, to calculate atomic concentrations from spectra and depth profiles, the beam sensitivities for each element must be determined for each instrument. The graphs of the standards in this document are derivatives of the intensity versus energy spectra with electron beam voltages varying between 3KV, 5KV, 10KV, or 15KV. In some instances, the samples were ion beam sputtered to remove surface layers. Sensitivities of each element to each beam voltage were determined and recorded on the table of properties noting differences between the calculated and published values.

## INTRODUCTION

This document is to be used as a reference of standard data for identification of Auger electron spectroscopy data for the Perkin Elmer Scanning Auger Multiprobe (PHI 600) at Rome Laboratory.

Auger electron spectroscopy is based on the Auger radiationless process. When an electron beam ionizes the core level of a surface atom, the atom transforms into a doubly ionized state. Each element has a particular transition energy which is transferred to the ejected Auger electron. This transition of energy produces a spectra characteristic of the particular element.

Spectra and beam sensitivities of general standards have been published previously. However, each Auger instrument varies slightly and that slight variation may cause an error in the identification and quantification of a certain sample. In doing this study, elements and compounds of known composition were examined by the Auger spectroscopy to determine peak intensities and sensitivities for different beam voltages.

The samples used in this investigation were mainly ones which are used to fabricate microelectronic devices. Spectra from each sample were acquired at beam voltages (BV) of 3KV, 5KV, 10KV, or 15KV. The beam current (BI) was varied depending on the sample and is noted on the graphs. Some samples were sputtered (SPU), using the ion beam to remove the surface layers or to neutralize charging effects. Sputtering during or



prior to acquisition is noted on the graphs. In order to obtain useful laboratory as well as technically correct data, the standards were acquired as if they were laboratory samples. Therefore, one must use care to correlate the acquisition parameters used on an unknown sample with those used on the standards. For example, if the analyzer was in the v/f mode for the standard then the analyzer must be in the v/f mode for the sample to be analyzed. The analyzer reacts differently in the v/f (analog) and p-c (pulse count) modes which changes the spectra characteristics.

Sensitivities were calculated at each beam voltage for each element using the formula  $(A + B / A) * I_x / I_{Ag} * K_x$ .  $I_x$  and  $I_{Ag}$  are the peak to peak amplitudes and  $K_x$  is the scale factor of the particular graph. A and B are chemical formula indices of compound  $XAYB$ . Where the sensitivities calculated differed from published values, the published values are noted at the bottom of the table.

The text consists of tables of properties of the sample studied. The tables do not include all of the properties but are, instead, designed to be a quick and easy reference of pertinent information. This document provides a sample of the summer research done, but does not exclusively cover everything. A Technical Memo will be published as a result of this summer research and will be available for further information.

Table 1: Antimony (Sb)

Table of Properties

Atomic Number	51	
Atomic Mass	121.75	amu
Melting Point (1 atm)	903.75	K
Boiling Point (1 atm)	2023.0	K
Specific Heat at 298K	0.049	cal/gK
Density	6.69	g/cm <sup>3</sup>
Entropy at 298K		
at 273.2K	0.255	W/cmK
at 298.3K	0.244	W/cmK
at 373.2K	0.219	W/cmK
Sensitivities		
at 3KV	0.6772	
at 5KV	1.0361*	
at 10KV	1.5225*	
at 15KV	1.0629	
Crystal Structure	rhombohedral	
Lattice Spacing	depends on polymorphic form	

\*Perkin Elmer values  
5KV=.65  
10KV=.4

Figure 1

AES SURVEY V/F 6/28/91 AREA 1

FILE: sb3.100d Original Fil d. '00d Massage

SCALE FACTOR= 376.341 k c/s, OFFSET= 0.0

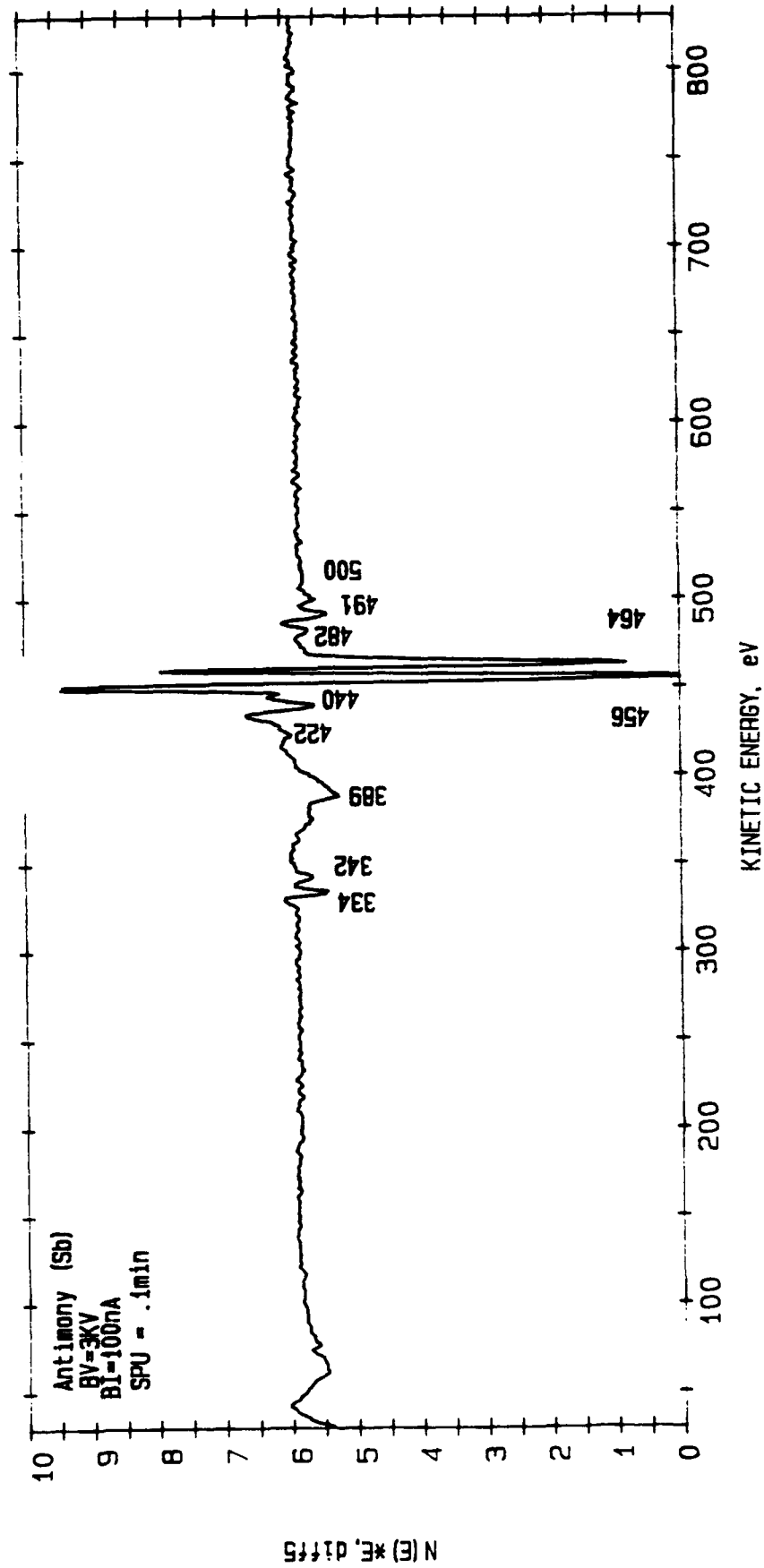
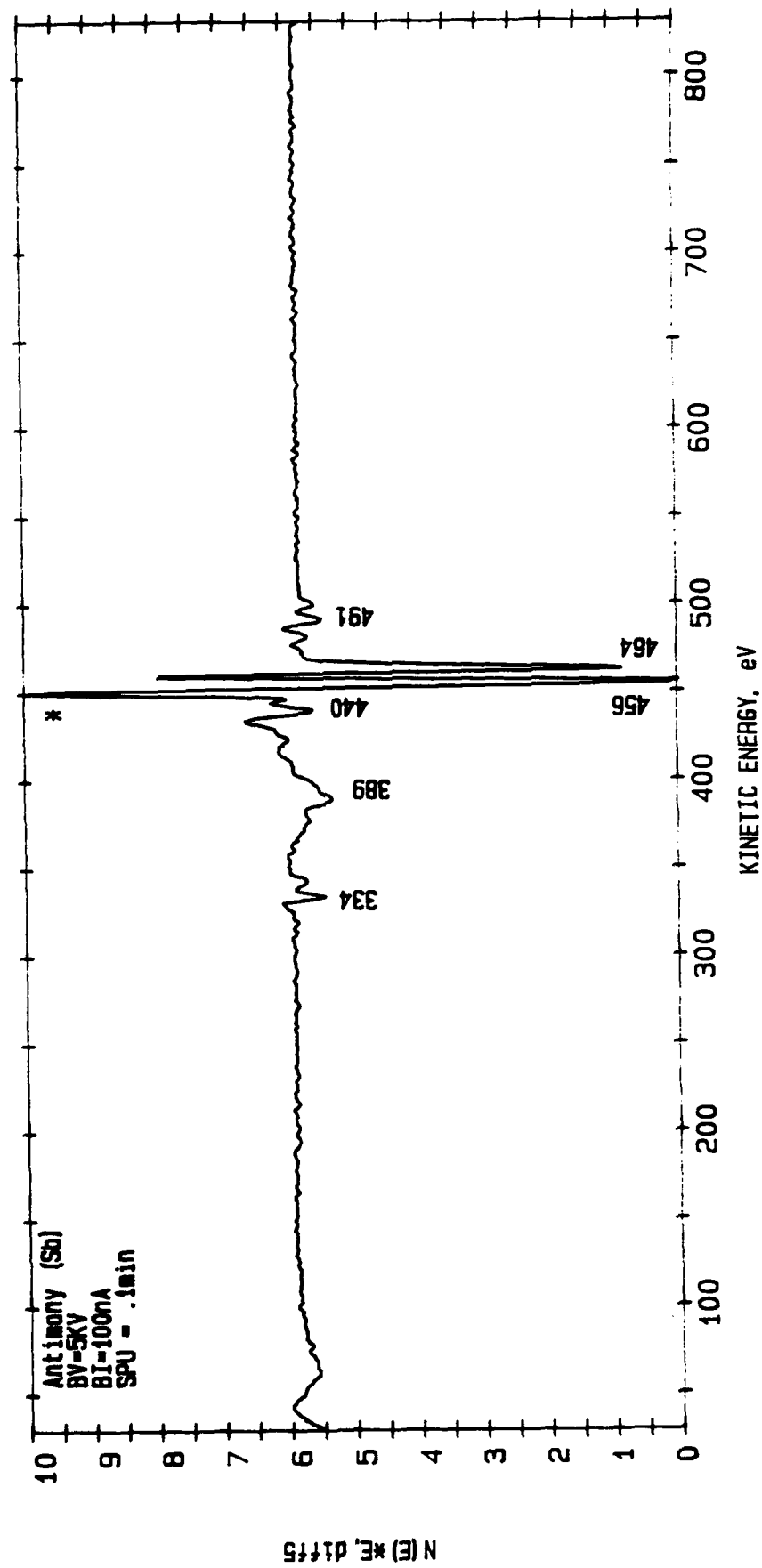


Figure 2

AES SURVEY V/F 6/28/91 AREA 1

FILE: sb5.100d Original Filename: sb5.100d Massage

SCALE FACTOR= 657.450 k c/s, OFFSET= 0.000 k c/s



- 1: Select Files
- 2: Receive files via Kermit
- 3: Convert/merge files
- 4: Perform FFT
- 5: Other
- 6: Exit

Input Files:

in1-095  
in2-095

Messages:

READY

Enter your choice: ☐

Figure 4

AES SURVEY P-C 6/28/91 AREA 1

FILE: sb15.100d Original Filename: sb15.100d Message

SCALE FACTOR= 4615.227 k c/s. OFFSET= 0.000 k c/s

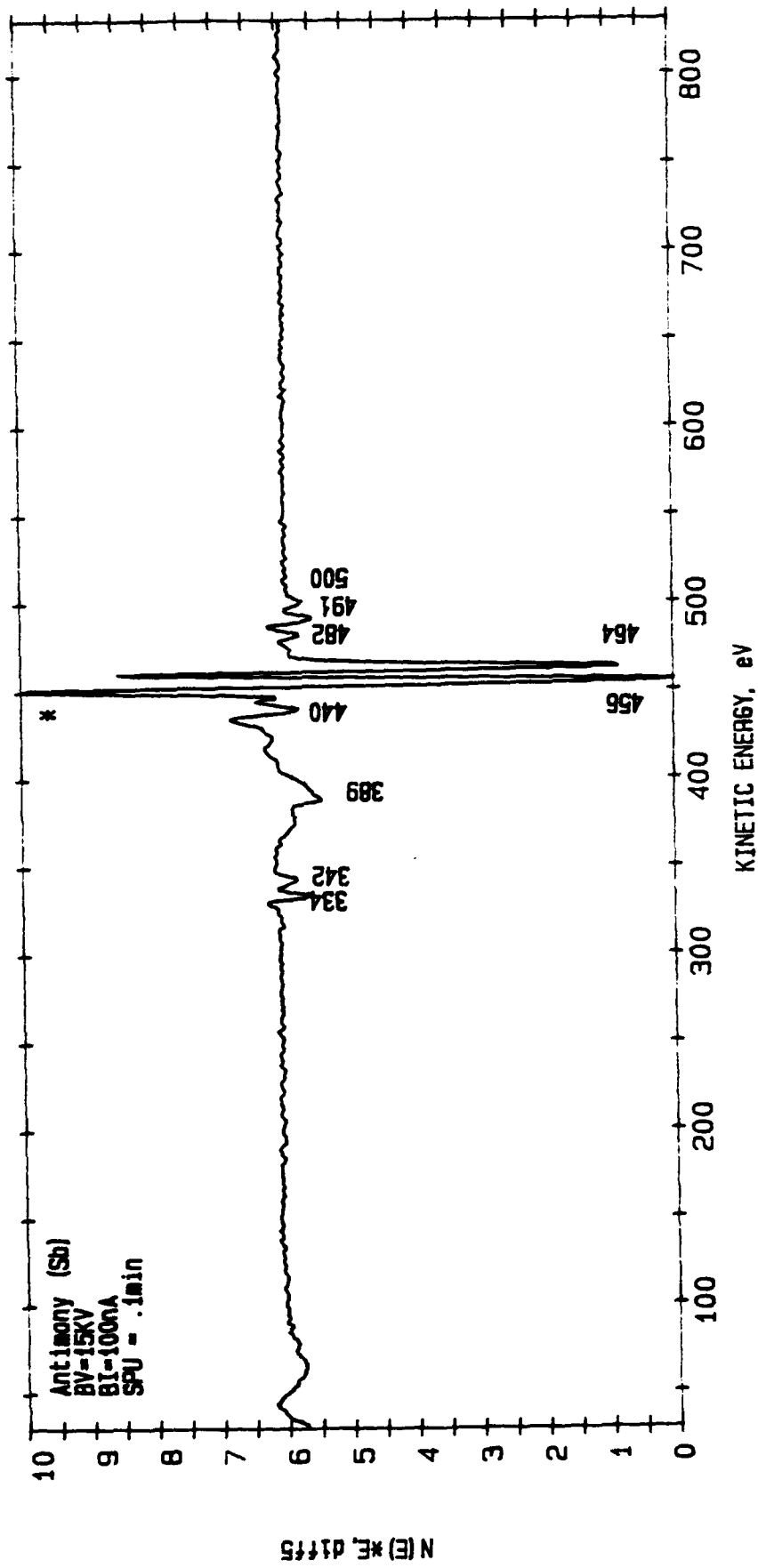


Table 4: Cadmium Sulphide (CdS)

Table of Properties

Molecular Weight	144.47	amu
Melting Point (100 atm)	2023.0	K
Boiling Point (1 atm)	1253.0	K
Density	4.82	g/cm <sup>3</sup>
Sensitivities		
at 3KV Cd	1.3139	
S	1.7917	
at 5KV Cd	1.5267	
S	2.2811	
at 10KV Cd	2.1011	
S	2.4681	
at 15KV Cd	1.3646	
S	1.5188	
Crystal Structure	both wurtzite and sphalerite	
Lattice Spacing	0.5832/a=.416nm c=.6756nm	
Band-Gap Width and Type at 300K	2.42	eV direct

Figure 5

AES SURVEY V/F 6/28/91 AREA 1

FILE: c3.100d Original Filename: c3.100d Message

SCALE FACTOR= 92.126 k c/s. OFFSET= 0.000 k c/s

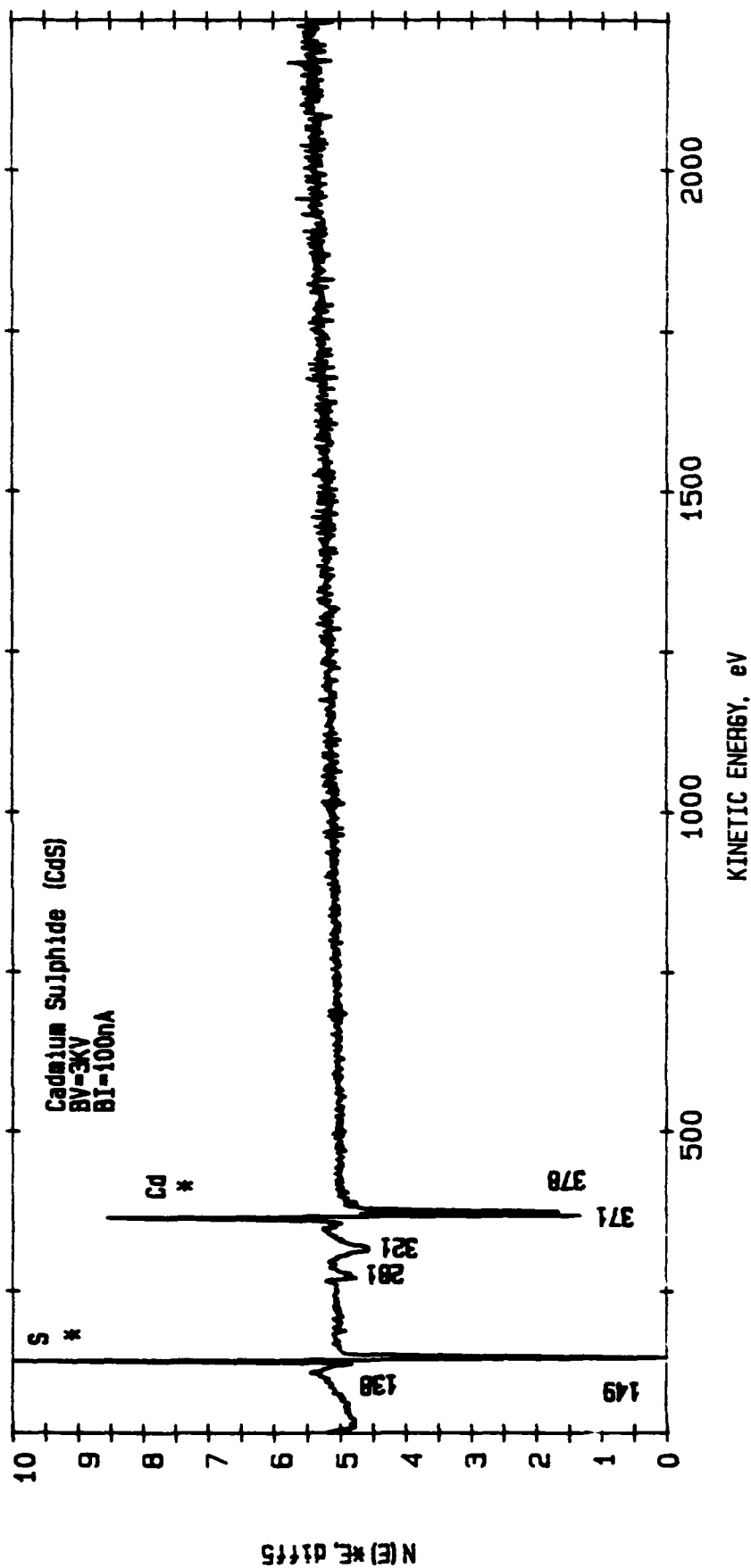




Figure 6

AES SURVEY V/F 6/28/91 AREA 1

FILE: cds5.100d Original Filename: cds5.100d Message

SCALE FACTOR= 696.928 k c/s. OFFSET= 0.000 k c/s

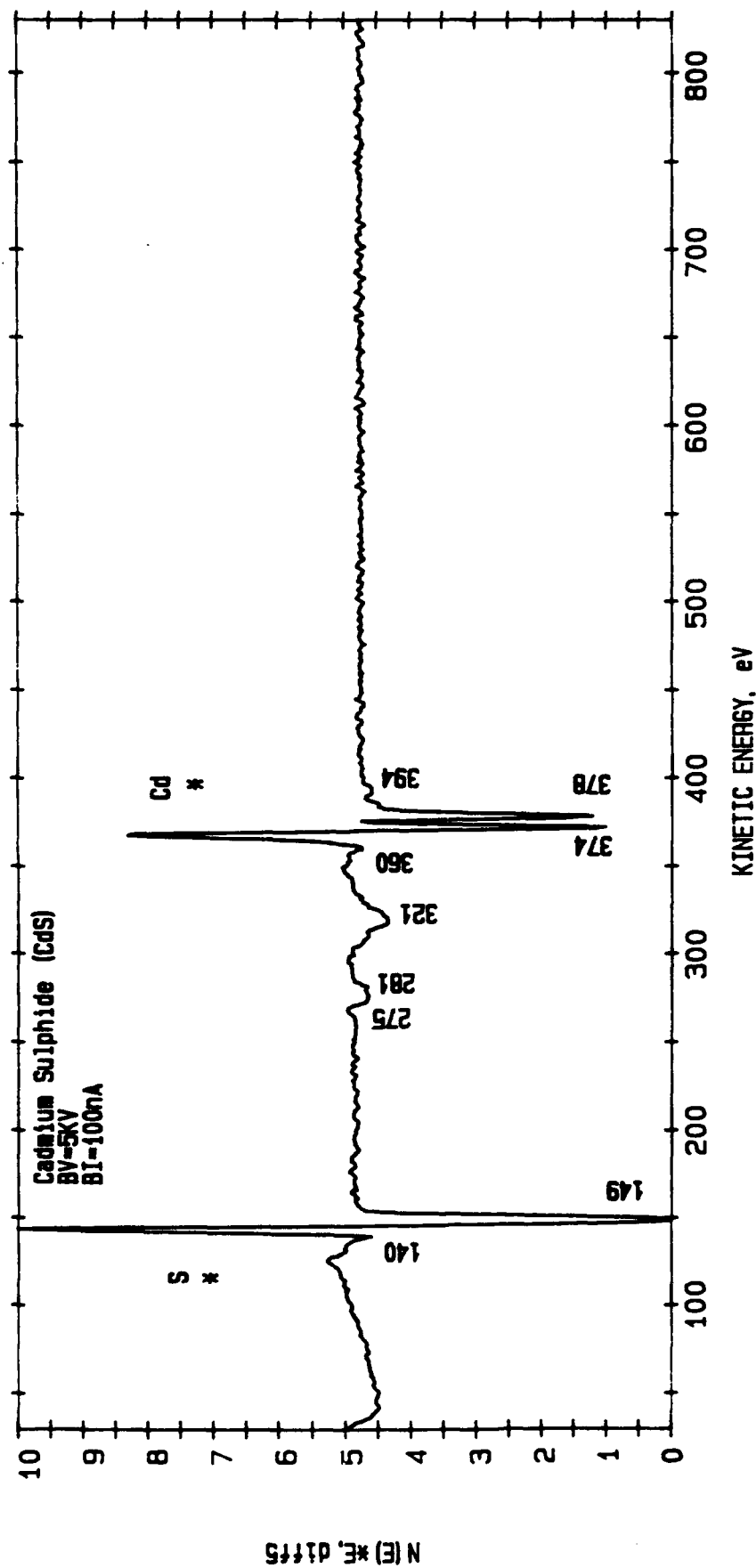


Figure 7

AES SURVEY P-C 6/28/91 AREA 1

FILE: cds10.100d Original Filename: cds10.100d Massage

SCALE FACTOR= 4158.725 k c/s. OFFSET= 0.000 k c/s

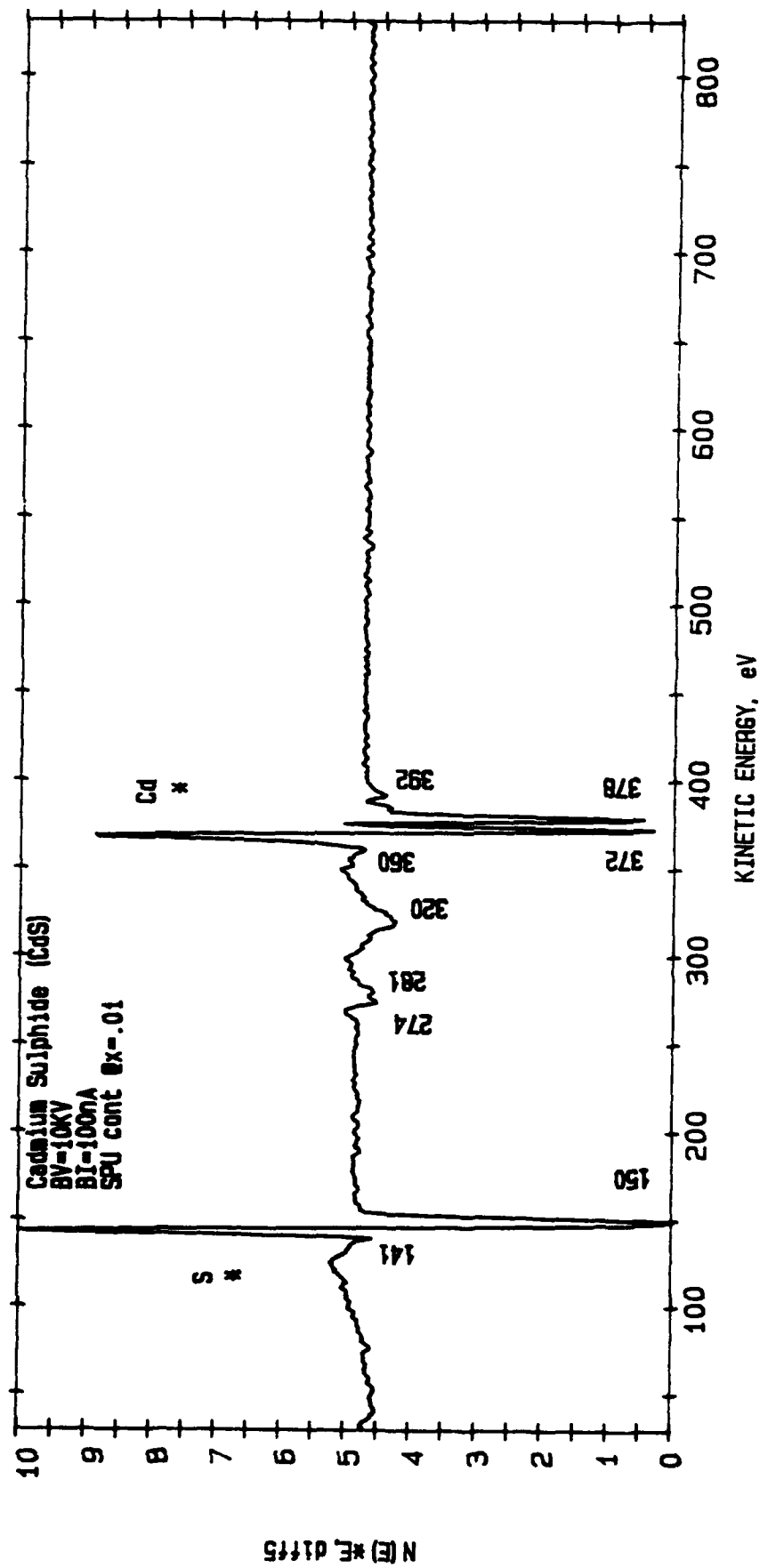


Figure 8

AES SURVEY P-C 6/28/91 AREA 1

FILE: cds15.100d Original Filename: cds15.100d Message

SCALE FACTOR= 3475.866 k c/s. OFFSET= 0.000 k c/s

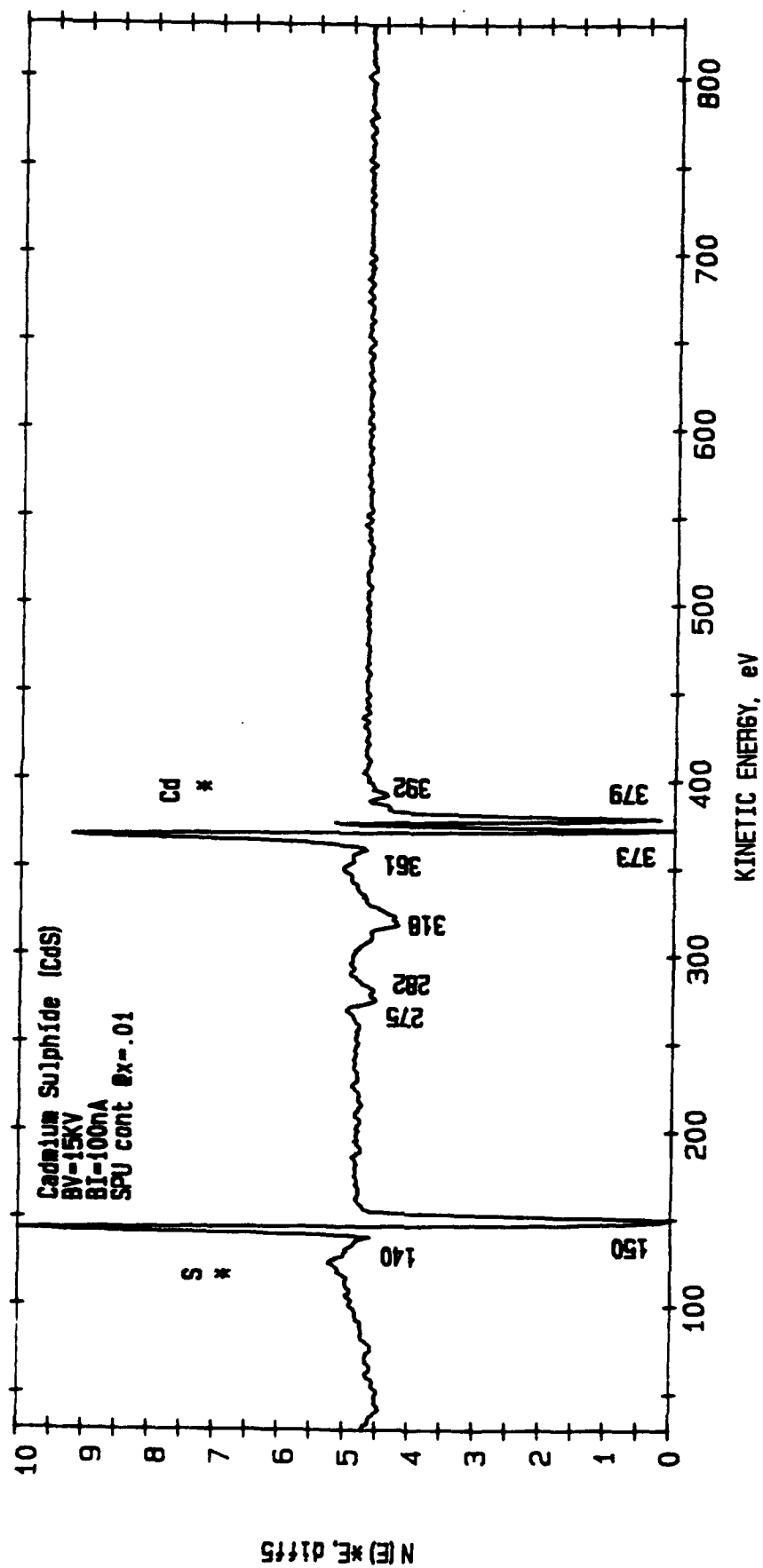


Table 24: Zinc Sulphide (ZnS)

<u>Table of Properties</u>		
Molecular Weight	97.44	amu
Melting Point (30 atm)	1973 $\pm$ 20	K
Boiling Point (1 atm)	1458	K
Density	4.079	g/cm <sup>3</sup>
Sensitivities		
at 3KV Zn	0.2173	
S	0.8621	
at 5KV Zn	0.3296	
S	1.0896	
at 10KV Zn	0.6254	
S	1.3277	
Crystal Structure	sphalerite or wurtzite	
Lattice Spacing	0.542/a = 0.382 nm c = 0.626 nm	
Band-Gap Width and Type at 300K	3.68 eV direct	

Figure 9

AES SURVEY V/F 6/28/91 AREA 1

FILE: zns3.100d Original Filename: ZNS100.0BJ.mac Survey Area 1

SCALE FACTOR= 227.317 k c/s. OFFSET= 0.000 k c/s

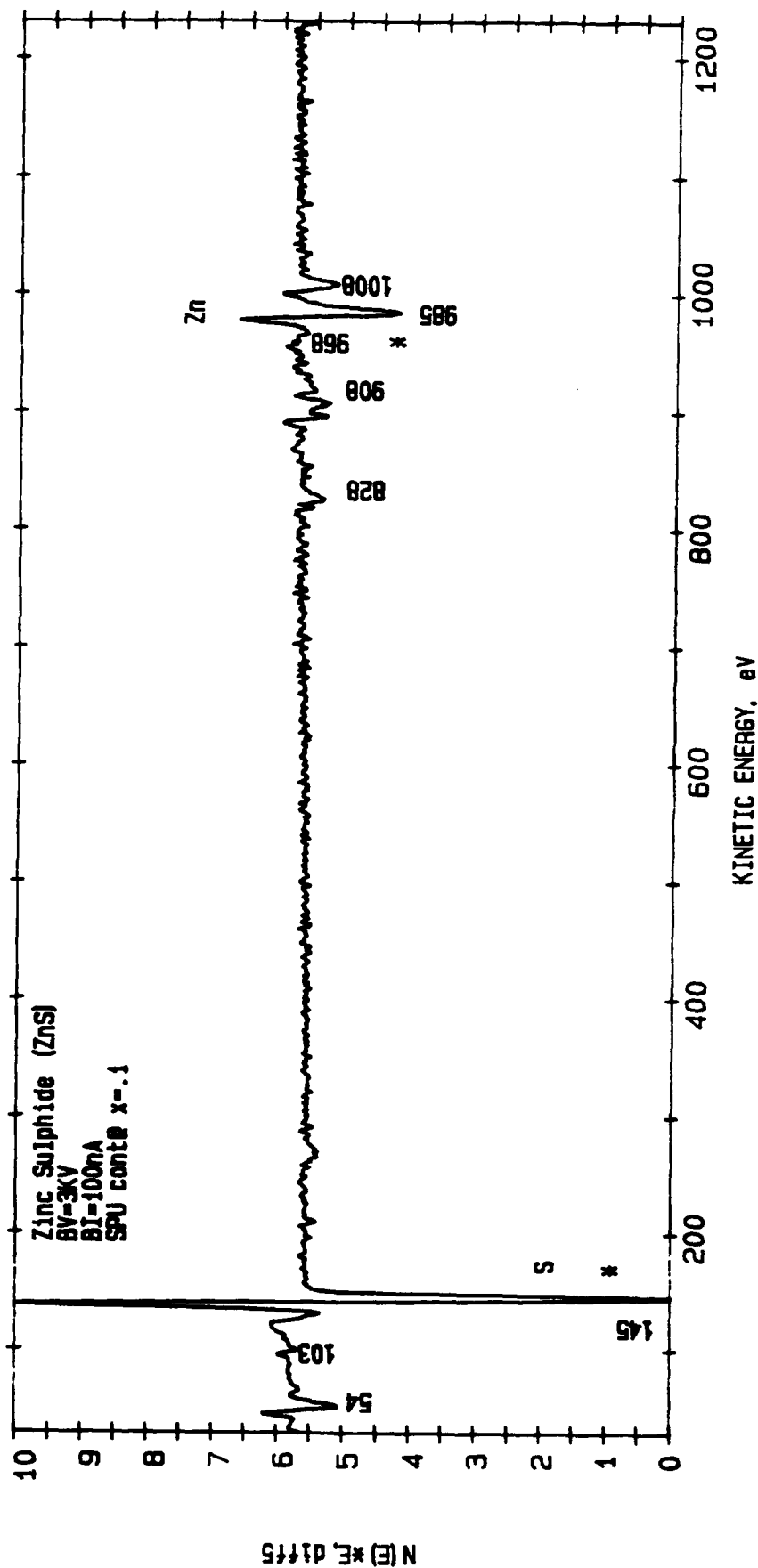


Figure 10

AES SURVEY V/F 6/28/91 AREA 1

FILE: zns5.100d Original Filename: zns5.100d Message

SCALE FACTOR= 320.051 k c/s, OFFSET= 0.000 k c/s

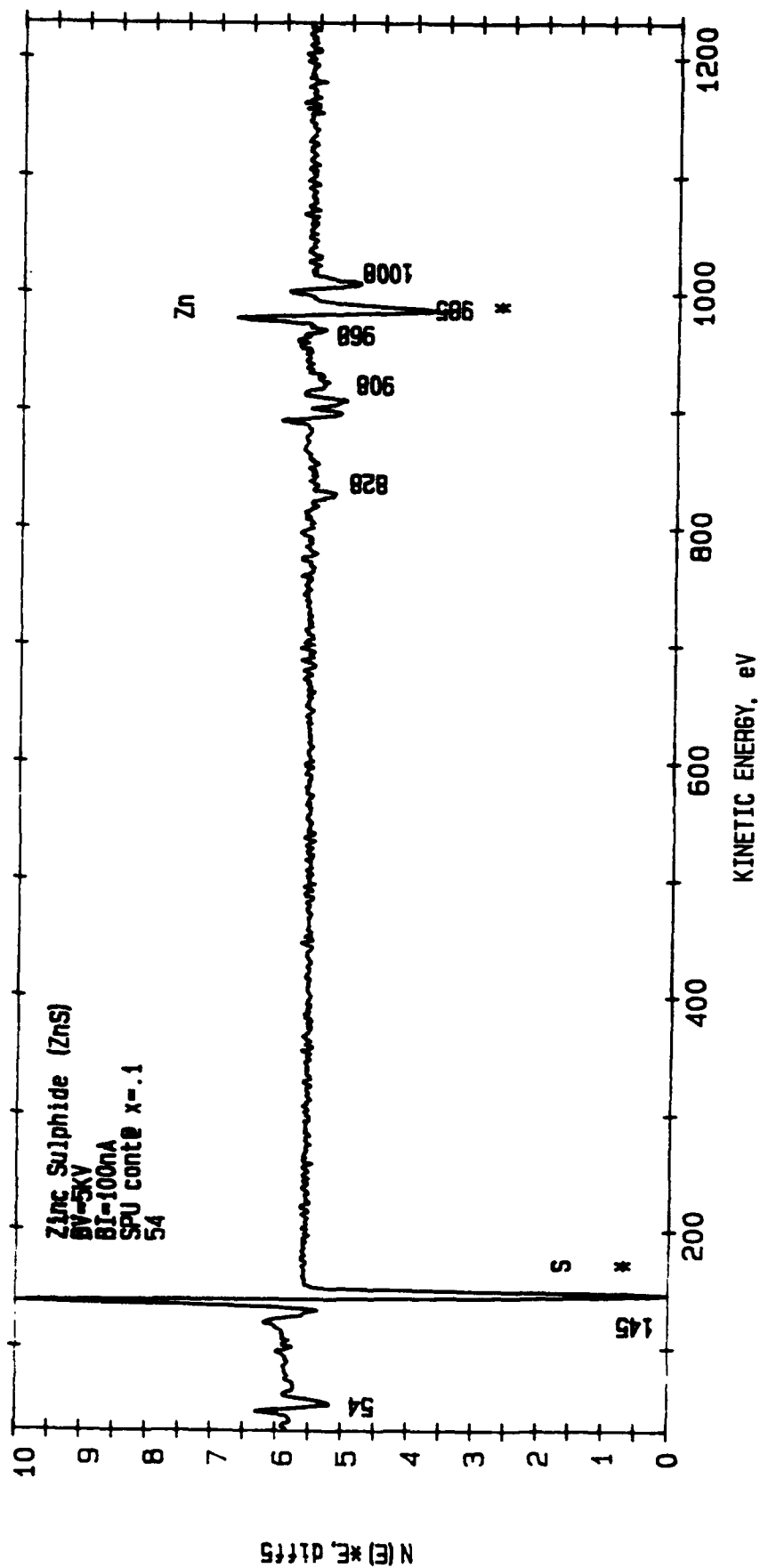
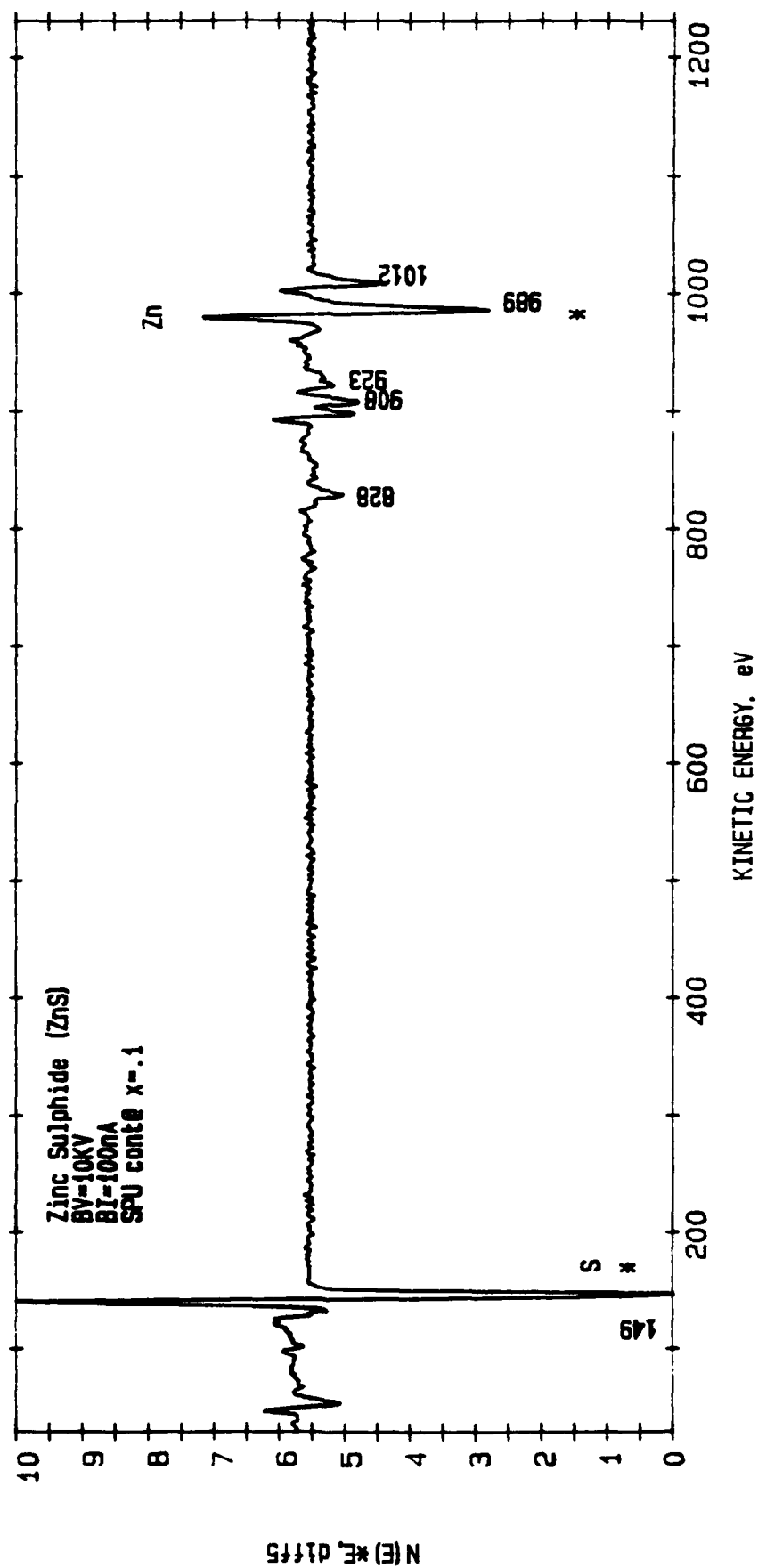


Figure 11

AES SURVEY V/F 6/28/91 AREA 1

FILE: zns10.100d Original Filename: zns10.100d Massage

SCALE FACTOR= 612.356 k c/s, OFFSET= 0.000 k c/s



## Bibliography

- Davis, Lawrence E., MacDonald, Noel C., Palmberg, Paul W., Riach, Gerald E., Weber, Roland E. Handbook of Auger Electron Spectroscopy. Physical Electronics Industries: Minnesota, 1976.
- Grovenor, CRM. Microelectronic Materials. Adam Hilger: Philadelphia, 1989.
- Kittel, C. Introduction to Solid State Physics Fifth Edition. John Wiley and Sons: New York, 1976.
- Maissel, Leon I. and Glang, Reinhard. Handbook of Thin Film Technology. McGraw-Hill Book Company: New York, 1970.
- West Ph.D, Robert C. CRC Handbook of Chemistry and Physics. CRC Press, Inc: Boca Raton, Florida, 1976.



## RDL SUMMER RESEARCH PROGRAM

Jason D. Kowalczyk

My summer research project entailed many small projects of different types. The major function of my apprenticeship was programming in 'C'. I also worked on the support of the ELINT Developmental Facility by doing computer maintenance and testing new hardware and software that the lab would be using in the coming months. Most of the testing was done on a IBM<sup>TM</sup> Compatible Personal Computer platform but some was using SPARCstation 1+<sup>TM</sup>, Macintosh<sup>TM</sup>, and DEC<sup>TM</sup> Products.

The main project was Hexdump Utility to validate that the network inside the lab was working properly and the large data files still had data integrity. One of the major problems with the "C" code was the input and the output of the program to and from the hard disk without using ANY DOS function calls or any IBM-PC function calls. This is to keep the Brian W. Kerighan and Dennis M. Ritchie standard. To have total code portability throughout the lab.

The problem was hard to get around due to the fact I use all of the standard DOS and IBM-PC function calls to get input and output to and from the disk. Some of the functions work only with IBM-PC's and compatible including:

```
long lseek(int handle, long offset, int fromwhere);
```

```
int open(const char *pathname, int access [, unsigned mode]);
```

```
int read(int handle, void *buf, unsigned len);
```

```
int write(int handle, void *buf, unsigned nbyte);
```

Even though these code examples are for DOS and/or IBM-PC platforms there are some code examples for the Brian W. Kerighan and Dennis M. Ritchie and ANSI C standard. The only way to keep the code portability is to include the following statements:

```
int fseek(FILE *fp, long offset, int whence);
```

```
FILE *fopen(const char *filename, const char *mode);
```

```
int fgetc(FILE *fp);
```

```
int fputc(int c, FILE *fp);
```

The results after debugging the program came out like this:

```

/*****
***          SEI Database Hexadecimal Dump Utility          ***
***
***          Executed using the IBM-PC compatible Platform  ***
***
***          Programmed by : Jason D. Kowalczyk             ***
***                      ROME LABORATORY                   ***
***                      GAFB                               ***
***                      ROME, NY                          ***
***                      (315) 330-7557 Elitent Dev. Fal.    ***
***                      (315) 330-4581 EDF/IRAP            ***
***
*****/

#include <stdio.h>
#include <io.h>
#include <fcntl.h>
#include <sys/stat.h>

#define X_LINE 8
#define LINELENGTH 16

main()
(
/*****
***          Declarations          ***
*****/

    char infile[30], outfile[30];

/*****
***          Setting Streams          ***
*****/

    FILE *in;
    FILE *out;

/*****
***          Declares and Sets Variables to ZERO          ***
*****/

    int ch = 0;
    unsigned int char_dumped = 0;
    unsigned int seek = 0;
    unsigned int stop = 0;
    unsigned int total = 0;
    int linecount = 0;
    unsigned int def = 0;
    int total_2;

/*****
***          Declares and Sets Variable Value          ***
*****/

    int length = 1;
    int t_char = 1;

/*****
***          Gets INPUT file NAME          ***
*****/

    printf("\n\nPlease Enter 'INPUT' File Name : ");

```

```

scanf("%s", infile);

/*****
***      Opens the INPUT file and checks for      ***
***      it to exist and for READ BINARY access    ***
*****/

if ((in = fopen(infile, "rb")) == NULL)
{
    printf("\n\nERROR !! --> Cannot open INPUT file.\n");
    return (1);
};

/*****
***      Gets OUTPUT file NAME                    ***
*****/

printf("\n\nPlease input 'OUTPUT' file name : ");
scanf("%s", outfile);

/*****
***      Opens the OUTPUT file. For APPEND BINARY access ***
*****/

if ((out = fopen(outfile, "ab")) == NULL)
{
    printf("\n\nERROR !! --> Cannot open OUTPUT file.\n");
    return (1);
};

/*****
***      Gets Starting Sample number              ***
*****/

printf("\n\nPlease Enter starting sample : ");
scanf("%d", &seek);

/*****
***      Gets Ending Sample number                ***
*****/

printf("\n\nPlease Enter ending sample : ");
scanf("%d", &stop);

/*****
***      Checks user's input for The reverseing    ***
***      of START and STOP sample numbers          ***
*****/

while ( stop < seek ) (
    printf("\n\nYou Can not list data samples Backwards");
    exit(0);
);

/*****
***      Computes the total number of samples to be printed ***
***      +1 is to fix the fseek parameter because the offset is ***
***      '0' to read in the first byte              ***
*****/

total = stop - seek +1;
total_2 = total;

```

```

/*****
*****
*****
***      Now the program Starts to dump Characters to the OUTFILE      ***
*****
*****
*****/

/*****
***      Prints the Header On the top of the SCREEN      ***
*****/

printf("\n\nSEI Database Hexadecimal Dump Utility\n");
printf("Executed on the IBM-PC compatible Platform\n\n");
printf("Input File   : %s\n", infile);
printf("Output File  : %s\n", outfile);
printf("Total samples : %d\n", total);
printf("Samples from %d to %d\n\n", seek, stop);

/*****
***      Prints how the HEX DUMP should be read since the VAX hex dump is ***
***      Exactly opposite of the IBM-PC platform      ***
*****/

printf("v-- Sample #1                      Sample #16 --v\n\n");

/*****
***      Prints the Header in the OUTFILE      ***
*****/

fprintf(out, "\r\n\nSEI Database Hexadecimal Dump Utility\r\n");
fprintf(out, "Executed on the IBM-PC compatible Platform\r\n\n");
fprintf(out, "Input File   : %s\r\n", infile);
fprintf(out, "Output File  : %s\r\n", outfile);
fprintf(out, "Total samples : %d\r\n", total);
fprintf(out, "Samples from %d to %d", seek, stop);
fprintf(out, "\r\n\n");

/*****
***      Prints how the HEX DUMP should be read since the VAX hex dump is ***
***      Exactly opposite of the IBM-PC platform      ***
*****/

fprintf(out, "v-- Sample #1                      Sample #16 --v\r\n\n");

/*****
***      Changes the file pointer location to access the right sample      ***
*****/

fseek(in, (seek-1), 0);

/*****
***      LOOP TO START HEX DUMPING TO BOTH THE STDOUT AND OUTFILE      ***
***      Checks for the End of File Marker      ***
*****/

while (ch != EOF) {

/*****
***      Checks to see if it needs to stop dumping HEX      ***
*****/

    if (total == char_dumped) {
        printf("\n\nByte Dump Complete\n\n");
        fprintf(out, "\r\n\nByte Dump Complete\r\n\n");
        exit(0);
    };

/*****
*****/

```

```

***          Gets ONE character at the current pointer location          ***
*****/

        ch = fgetc(in);

/*****/
***    Prints that character into the STDOUT, and formats it with %X to    ***
***    output it as a HEX with Capitol Letters                            ***
*****/

        printf ("%2X ",ch);

/*****/
***    Prints that character into the OUTFILE, and formats it with %X to    ***
***    output it as a HEX with Capitol Letters                            ***
*****/

        fprintf(out,"%2X ", ch);

/*****/
***    Checks if it needs to drop the next line for formating reasons    ***
*****/

        if (length == LINELENGTH) {
            printf("\n\r");
            fprintf(out,"\n\r");
            length = 0;
            linecount = linecount +1;

/*****/
***    checks if the consecutive lines are = to '8'                        ***
***    so it can drop 2 lines instead of 1                                ***
*****/

            if (linecount == X_LINE) {
                printf("\n");
                fprintf(out,"\r\n");
                linecount = 0;
            }

        };

/*****/
***          UPDATES COUNTERS BY "1"          ***
*****/

        length = length + 1;
        char_dumped = char_dumped + 1;
        t_char = t_char + 1;

    };

/*****/
***          Prints total # of Samples outputed                          ***
*****/

        def = t_char - total;

        printf ("\n\nTotal Samples Outputed before EOF Marker : %d",t_char);
        printf ("\nTotal Samples Requested                      : %d",total);
        printf ("\nA Defichiancy of %d Samples", def);

        fprintf (out,"\r\n\nTotal Samples Outputed before EOF Marker : %d",t_char);
        fprintf (out,"\r\nTotal Samples Requested                      : %d",total_2);
        fprintf (out,"\r\nA Defichiancy of %d Samples", def);

/*****/
***          CLOSES ALL OPEN FILES          ***
*****/

        fclose (in);

```

```
    fclose(out);  
}
```

I've come across many difficulties during my research project. Some of these include computer "glitches" in Turbo C<sup>TM</sup> and DOS<sup>TM</sup>. Halted programming for several days. Even though I've had some setbacks during the summer, I've accomplished many of the small projects around the lab. Some of the small projects were the installing and testing of the CD-ROM disks and the testing of new anti-virus programs from Central Point<sup>TM</sup> and other small pieces of software around the lab. I also spent time using the Internet facilities to find "C" source code for office another project.

In conclusion, this summer was a very enlightening experience. Even though I already knew the "C" language, I learned how to use the file I/O in "C" using the Brian W. Kerighan and Dennis M. Ritchie standard. I would now like to take the time to thank all of the people at Rome Laboratory for the help in "C" programming. I would like to especially thank John Pletl, Jeffery Klemczyk, and Mike Symanski for all of the guidance during the summer. I would like to thank the people at Research & Development Laboratories for selecting me for this position. I thank you for the experience and I would like to express my interest in participating next year.

**GEMACS REVISIONS AND UTILIZATION**

**Jason C. Lenio**

**Air Force Office Scientific Research  
High School Apprenticeship Program  
Research and Development Laboratories**

**Rome Labs  
Air Force Systems Command  
Griffiss AFB, NY 13441-5700**



#### ACKNOWLEDGEMENTS

I would like to thank John Cleary, Ken Siarkiewicz, Mike Seifert, Anthony Pesta, Paul DiCarlo, and Timothy Blocher for their instruction and benevolence over the summer.

## 1.0 INTRODUCTION

This report describes the work done by the High School Apprentice in ERPT during the summer of 1991. This work was primarily concerned with the revision and utilization of the Electromagnetic Environmental Effects (E3) program, General Electromagnetic Model for the Analysis of Complex Systems (GEMACS). The tasks regarding the GEMACS program consisted of generating a release version of the program, debugging the code, the validation of test cases, and producing simulation models for GEMACS analyses using the Graphical Aids for the Users of GEMACS (GAUGE) tool.

In Versions 5.0-13 and 5.1 GEMACS program, certain statements can only be understood by running the VMS operating system. These statements needed to be replaced by equivalent language to allow compatibility with other systems. Once a releasable version of the program was generated, test cases were run for assurance that the version ran correctly.

A model of a simple cavity with a loaded wire inside was generated by using the Graphical Aid for the Users of GEMACS (GAUGE) tool. The cavity was to simulate a cavity of an aircraft which contained cables leading to electronic equipment. This model was to be tested for susceptibility to electromagnetic threats, using the new version of GEMACS. A simple dipole wave source was placed five meters in front of the cavity. The wire was then submitted to GEMACS without the geometry of the cavity. The outputs of these two tests were to be compared to determine the difference between the currents produced on the wires inside and outside of the cavity when struck by incident electromagnetic radiation.

## 2.0 GEMACS

### 2.1 OVERVIEW

GEMACS is a user oriented, general purpose code, which is designed to incorporate many techniques in simulating complex electromagnetic systems. The program is designed for experienced electromagnetics analysts, with knowledge of applied linear algebra [1].

The program consists of six executable FORTRAN programs. These programs, called modules, are Geometrical Theory of Diffraction (GTD), Method of Moments (MOM), Finite Differences (FD), Input, Solution, and Output. The GEMACS code uses high level language and provides flexibility over the computational sequences, for ease to the user. The program also contains many debug features to help the user in identifying and locating errors.

### 2.2 GENERATING RELEASE VERSIONS OF GEMACS

Recently versions 5.0-13 and 5.1 of the GEMACS program were delivered to the Rome Labs (RL). Their increased capabilities and capacities are currently being assessed at RL/ERPT. These versions have very portable codes, but the editions given to ERPT are designed to run on a DEC MicroVAX system. In various subroutines of the programs there are INCLUDE statements which refer to often called upon command strings, known as common blocks. Include statements allow each command to be written only once per subroutine and then referred to multiple times. This shortens the code and makes it much easier to read and write. Some computers, other than a VAX system, may not be able to process this statement. If the GEMACS program is to be distributed to Department of Defense (DoD) contractors for use, it must be able to run on a variety of systems. To fix this, the include statements must be replaced by the common block that

they represent. The task is simple, yet rather tedious.

This chore takes nearly two weeks to complete and must be performed each time ERPT receives an updated version of GEMACS. It became evident that a program should be developed that would emulate the manual revision. This program was to be written using the DIGITAL Command Language (DCL). The DIGITAL Command Language is a limited set of English-like instructions that tell the VMS operating system to perform specific operations. [2] After extensive studies, it was concluded that the DCL was unable to accomplish the objective. However, using the DCL, a small program was created that would fulfill a fraction of the modifications. This succinct program reduced the manual revision time to approximately one week.

Once the files were edited they then had to be recompiled. The files were compiled using a batch file. Batch files are noninteractive processes that perform the desired task in the background [3]. This allows the user to execute other tasks at the same time. This saves a large amount of time. If there were no errors, an object file was created from it. If there was an error, then no object file was created. To correct the error the user must examine the .LIS file. Compile errors are denoted in the .LIS files, making the program much easier to use. When the error was found, the FORTRAN file was then repaired and recompiled. Usually the error consisted of no more than a typographical error, or an include statement that had been forgotten. Using another batch file, a library was created in each module. Then the object files were linked, using yet another batch file. From the linked files an executable was generated for each module. The .EXE files were now ready for testing.

### 2.3 PROBLEMS WITH VALIDATION

To insure that these release versions were working

properly, test cases, which had been previously validated, were run through the programs. The cases varied from open ended waveguides to impedance loaded arrays of quarterwave dipoles. Three cases, WAVEGUIDE, CAVITY, and THIELE, were submitted to each version of GEMACS that had been edited.

When the sample problems were executed in GEMACS 5.0-13, WAVEGUIDE and THIELE, received error messages. Suspect files were then re-edited. This did not rectify the error. The next step taken was to re-edit entire modules that were furnishing errors. After re-editing the SOLUTION and OUTPUT modules, WAVEGUIDE and THIELE were resubmitted. Upon completion, only WAVEGUIDE contained an error. This error was searched for periodically over the summer and could not be located. Therefore the revised version of GEMACS 5.0-13 was considered not valid.

In the execution of the sample problems in GEMACS version 5.1, an error was received in the THIELE example. This error was carefully investigated and was soon discovered. A common block had been omitted from a file. This was repaired and the sample problem was resubmitted and executed without error. The results of the examples were obtained and then compared to the results of the same tests run through the unedited edition of GEMACS 5.1 delivered to RL/ERPT. The outputs were the same, therefore the edited version of GEMACS 5.1 was considered valid and was ready for distribution.

### 3.0 ELECTROMAGNETIC SIMULATION

#### 3.1 OVERVIEW OF GAUGE PROGRAM

Graphical Aids for the Users of GEMACS (GAUGE), is a program designed to provide a graphical interface for GEMACS users [4]. The GAUGE program allows the user to develop and visualize geometry input for GEMACS. GAUGE is written in the FORTRAN 77 language, except for a few

auxiliary files, which are written in Turbo Pascal. The program also allows the user to view color a representation of variations in electrical attributes of structures, such as impedances and currents.

### 3.2 OBJECTIVES OF SIMULATIONS

The task, using GEMACS and GAUGE, was to develop and simulate a box containing a wire, that is exposed to an electromagnetic field. This model mimics an aperture on the cavity of an aircraft containing a cable. A second model was generated consisting of just the wire. These models were created to analyze the effect that electromagnetic radiation coupling on to cables has on the electronics connected to the cables. This technique to determine the transfer function between the exterior and interior of the box will be very useful when calculating the interfering signal conducted into a circuit for circuit simulations.

### 3.3 MODEL CONSTRUCTION

Although the GAUGE/GEMACS interaction is a versatile one, it does have its limitations. GEMACS requires that finite difference cells be no larger than .1 wavelength in length each side. The wavelength is obtained through the following formula.

$$\text{Lambda} = c/f$$

Lambda = wavelength in meters

c = speed of light (3.0 E08 m/sec)

f = frequency in hertz

The cases were to be run at frequency of 1 GHz and as one can see, this does not allow any side of the finite difference cell to be more than three centimeters in length. The original size of the box that was to be used was 30cm x 60cm x 90cm. This box would produce a minimum of 6,000

finite difference cells. A similar problem containing approximately 2,000 cells required over two weeks of runtime on the fastest computer available. Thus, it became apparent that a smaller box must be utilized. The box was reduced to 15cm x 18cm x 21cm, which produced 210 cells. This reduced box demanded only 28 hours of runtime.

### 3.4 INPUT DECK FOR GEMACS

Before GEMACS can understand the GAUGE geometry, a list of commands, called the input deck, must be formed. In the input deck the user describes to GEMACS the specific details of the simulation that is to be run. A dipole wave source was placed 5 meters in front of the box. The dipole emitted a frequency of 1 GHz. The input deck consisted of two different regions, the exterior and the interior. The interior was comprised of finite difference cells, a loaded wire with an impedance of 50 ohms, and an aperture. The exterior was made up of plates that formed a box and an aperture. The apertures of each region had to be connected in order for GEMACS to acknowledge them. Once the once the input deck was formed the problem was ready for submission.

### 3.5 PROBLEMS ENCOUNTERED

Many errors were received in attempting to run these problems. The box problem received a negative logarithm error when it was first run. The cause of this error was not easily detected. The first step was to examine the commands in the input deck for flaws. When no errors were found, the next step was to investigate the geometry of the box for any violations. When no violations could be detected, assistance from local GEMACS authorities was called upon, and the documentation to GEMACS was examined. The error pinpointed the subroutine TLCELL, which dealt with interactions between cells and wires. After researching TLCELL

it was discovered that finite difference cells cut wire segments into further divisions called seglets. These cell boundaries had been cutting the wire longitudinally in half making it impossible to receive calculations. To correct this, the wire was moved slightly lower in the box. The box problem was then resubmitted. With this revision the box executed without error and output was obtained.

The model of the wire was then submitted. The setup was identical to the box except for the absence of the exterior region. This problem was very simple, yet it would not run to completion and gave many assorted errors. The errors could not be located and research on this problem is continuing at ERPT.

#### 4.0 CONCLUSIONS

Many things were accomplished during the summer apprenticeship. A release version of GEMACS 5.1 was produced. Experience was gained in the programming and use of the DIGITAL Command Language. A greater understanding of the GEMACS and GAUGE programs was achieved. Perhaps the greatest result was learning how to deal with problems that were encountered and investigating errors that were received. The research that was accomplished will be very beneficial to my future and to that of ERPT.



## 5.0 REFERENCES

1. E.L. Coffey, "GEMACS Version 5, Volume I - User's Manual", Draft Final Report, Advanced Electromagnetics.
2. "VMS Version 5.2 - User's Manual", Digital Equipment Corporation, June 89.
3. "VMS Version 5.2 - User's Manual", Digital Equipment Corporation, June 89.
4. A.J. Lockyer, P. Tulyathan, E.L. Coffey, M.J. Grage and G. Upshaw "GAUGE - User's Manual", RADC-TR-88-316, Northrup Corporation, Feb 89.

# **Artificial Neural Networks**

**Rome Laboratory**

**Rome, New York**

**by**

**Sean Menge**

## ABSTRACT

Research was performed over eight weeks to assess the use of artificial neural networks for computing versus using Conventional computers. Neural networks or neural "nets" are modeled after the human brain. The human brain contains billions of neurons which are connected to thousands of other neurons. The neural net contains many simple units called neurons, because of the way they function similar to the brain. Conventional computers, such as personal computers, mainframes ect., operate by changing numbers, symbols, and letters in a uniformed manner. Conventional computers can do many things such as add large numbers quickly or balance a persons budget. Although these computers can do work with little effort and time put into, conventional computers cannot do human-like functions. These functions include associative memory, vision, pattern recognition, and speech. For example, a conventional computer can add very large numbers in under one second but cannot identify a person from a photograph. In using PDP programs it was noticed that the answers were picked much like our own human brain would do it. Although the answer was not always right ninety percent of the time it was. With conventional computers the answer was right all the time but the neural net system was more efficient. Artificial neural networks are still in infancy meaning not too much is known on how they work. Programs were run on the ZENITH DATA SYSTEM using PDP software. The future for neural networks is very bright as commercial and military applications are starting to be seen.

## APPLICATIONS OF NEURAL COMPUTING

Neural networks model the way the brain encodes and processes information. They also are able to recognize pictures, teach themselves to read just like little children, speak, learn from past experience (memory), and perform a variety of pattern recognition tasks that conventional computers cannot do. Today applications of neural networks fall under three categories 1.) data analysis 2.) control systems and 3.) pattern recognition. The financial industry uses neural nets to assess problems with credit line usage, loan applications, and credit applications. The system is given relevant information from an applicant's form, it then decides from previous examples whether or not to accept or decline. Neural networks can also be applied to very large databases.

The second biggest application of neural networks is pattern recognition. Pattern recognition can be applied to speech recognition, industrial inspection, target recognition, medical imaging and also image analysis which includes seismic data, earth satellite images, and handwritten character reading. Recognizing human speech is a challenge that makes the computer identify speech, convert it to phonetic representations, and translate it into written text. Other promising applications include: radar, sonar, electronic warfare signal identification, adaptive systems ranging from flight to manufacturing controls, image compression, detection of strategic weapons by means of satellite sensors, stealth aircraft detection by infrared search-and-track systems, and photo interpretation.

Neural networks offer the opportunity of training a human level of perception and pattern recognition into a computer system. This is all

possible because the net, which mimics the actions of human brain cells, can learn and adapt by themselves. The network can easily switch from one task to another because it is not so much programmed as it is trained. Since neural networks are not programmed, then, there is no need for software or software development in the forms in which we have grown used to. Therefore development time and cost is lower, and networks can be implemented in simple hardware.

Neural networks have many important features and advantages, for distributed computing. In a local representation network, nodes do not have a simple meaning, but rather an individual concept is represented by a pattern of nodes. This offers the advantages of automotive generalization and insensitivity to damage. In a local representation if a system loses a node representing "dog" for an example, it loses the concept of "dog". In a distributed nature, in order to lose a concept the nodes representing it must be lost, This has lead to the assumption that the brain has a distributed nature.

Interconnections among the networks many processors, linked like nerve cells, are modified so that a particular input will produce a desired result. The computation is spread over many connections thus the network related an input pattern to an output pattern in a statistical rather than exact manner. The network can therefore process incomplete data, such as half of an image, just as the human brain can

do. A network can also recognize patterns despite noise(distorted).

Neural computers produce answers that are not always the very best, but are pretty good. With some tasks perfection may not be worth the extra time and effort spent, especially if there are good answers that can be found quickly. Absolute accuracy may not always be ideal. Reaching a good working solution fast rather than struggling for a long time for the best idea-may be the best way for finding routes for pipe lines, for example. Speed would also be more important than perfection for machines designed to recognize patterns and make generalizations. (Allman, "Desinging Computers" p. 62-63) Because computing can be done simultaneously by massive parallelism, tasks can be done much faster by neural computers than by conventional computers.

The benefits of artificial neural networks include: reduced design time as compared to rule-based systems, adaptive, trainable, naturally massively parallel, and highly fault tolerant systems. Also neural "nets" do not crash like conventional computers when damaged, instead they deteriorate and can be fixed by relearning.

The disadvantages and problems of neural networks are far outweighed by advantages. Also like humans the neural computer makes decisions based on past expirences rather than to follow rigid instructions. But often neural nets can not "explain" why they arrived at a given answer. This limits their usefulness, because many people can not accept an answer from a computer.

main memory. Memories consist of a large number of cells each one capable of storing a small amount of information.

Artificial neural "nets" are composed of simple processing units with weighted connections, arranged in layers. The more layers the more easily it can solve a task. These multi-layer "nets" categorize units into three classes 1.)input units-receive pattern direction 2.)output units-have associated teaching or target inputs and 3.)hidden units-neither receive input nor give direct feedback. These processing units function differently than those found in conventional computers. Conventional computers are either "on" or "off" with no in between position. Neural elements act like resistors and amplifiers. The output can vary between "on" and "off" because it multiplies inputs by weight factors. Outputs are the sum of the weighted input products.

Artificial networks can learn through a series of examples of pretenses on certain verbs, and eventually it will follow the changing rules for further similar verbs without being programmed with more examples. These rules follow the principle that if two units produce the right answer the weights between them will be increased. The reverse is true for wrong answers.

The artificial neural net can be trained in two ways, supervised

training and unsupervised training. In unsupervised training the training set consists solely of input vectors that are consistent. This training process takes properties of the training set and groups similar vectors into classes. To produce a specific vector one must apply a vector from a given class to an input. There are seven different tasks a neural net can perform, they are 1.)multi-sensing 2.)non-linear mapping 3.)sensory data processing 4.)self organization 5.)computational problems 6.)associative memory and 7.) classification.

CLASSIFICATION PARADIGM. The classification paradigm also can be considered as a variant on previous learning paradigms. There is a fixed set of categories into which the stimulus patterns are to be classified. There is a training session in which the system is presented with the categories to which each stimulus belongs. The goal is to learn to correctly classify the stimuli so that in the future when a particular stimulus or a slightly distorted version of one of the stimuli is presented, the system will classify it properly.

ASSOCIATIVE MEMORY Associative memory is just like that in humans. It is the ability to pick a face out of a crowd or so to speak. Associative memory is important to neural networks because conventional computers cannot give information back with only little input. Neural "nets" can give an output with little input and be right about ninety percent of the time. Associative memory is in fact when we learn to produce a particular pattern of activation on one set of units whenever another particular pattern occurs on another set of units. In general, such a learning scheme must allow an arbitrary pattern on one set of units to produce another arbitrary pattern on another different set of units. Associative learning is employed whenever we are concerned with storing patterns so that they can be re-evoked in the



future.

**NON-LINEARITY MAPPING** The use of non-linearity mapping occurs throughout with parallel distributed processing systems (Anderson et al., 1977; Grossberg, 1978; Hopfield, 1982; Kohonen, 1977). In figure 5 a perpendicular line was drawn to the weight vector. Since all vectors on this line project to the same point on the weight vector, their inner products within the weight vector are equal

Computational elements or nodes used in neural network models are non-linear are typically analog, and may be slower compared to modern digital circuitry. The simplset node sums  $N$  weighted inputs and passes the result through a non-linearity as shown in figure 6. The node is characterized by an internal threshold or offset  $\theta$  and by the type of non-linearity. Figure 6 illustrates three common types of non-linearities, hard limiters, threshold logic elements

**SELF-ORGANIZATION** Sensory systems have to go through brief critical periods in post-natal development in which the circuits are fine tuned in response to environmental exposure. Recent work by the Defense Advanced Research Projects Agency (DARPA) is determining the internal biochemical mechanisms. Network models are needed to explain how all these factors work together to produce the appropriate adult circuits, and there has been considerable work done in this area. In

addition to brief periods of self-organization during critical

periods, there is strong evidence for reorganization in adult brains as well recent work in the somatosensory system has demonstrated such processes of recovery at the neuronal level. The cortical topographic map of the skin is dynamically regulated by the differential use of skin regions. The cortical map territory representing a given skin region will shrink or expand as that skin region is stimulated less or more than neighboring skin regions, with consequent decreases or increases of sensitivity, respectively. Models seeking to explain how this self-regulation arises from neuronal mechanisms are clearly needed, and several have been proposed. Understanding of neuronal self-organization phenomena could lead to neural networks whose internal representations of sensory information would be maximally efficient and adaptive to changes in the environment.

**SENSORY DATA PROCESSING** An enormous amount of real time preprocessing is formed in the peripheral vision and hearing centers. Neural networks can perform this function in real time using massive parallelism

**COMPUTATIONAL PROBLEMS** Custom neural network architectures can be designed to solve specific computation problems, such as the "traveling salesman problem" and other constrained optimization problems, using nonlinear analog computation.

**MULTI-SENSING AUTOMATA** A number of complex, multi-module neural network automata have been built with visual input and a robot arm to manipulate objects in an environment. These automata demonstrate how an eye or camera can learn to scan a scene using self-supervision, and then how the eye and hand can be coordinated to perform simple tasks. These automata also demonstrate how inputs from multiple sensors can be fused

to provide classification performance better than could be achieved with a single sensor.

## MATERIALS

During the eight week research period two computers were used and one computer was referred to through past work. The ZENITH DATA SYSTEMS and the APPLE MACINTOSH PLUS were used during the research period while previous knowledge of the Commodore 64 computer was applied. Many simulations, programs, and references were made during research of artificial neural networks. References were made easy through use of the Rome Laboratory's technical library, where many other studies of neural networks were found. Also books, magazine clippings and DARPA's FINAL REPORT was supplied to me by my mentor Robert Kaminiski.

Programs run on the ZENITH DATA SYSTEMS computer with PDP software from David E. Rumelhart's and James L. McClelland's Explorations In Parallel Distributed Processing. Such programs were called on by PDP/.cs , PDP/.aa , PDP/.cl , PDP/.iaac , PDP/.ia , PDP/.bp , PDP/.pa . The program was divided into seven sections each dealing with a different type of neural network. The sections were as follows 1.)Constraint satisfaction (cs) 2.)The generalized delta rule (ia) 3.)The pattern associator (pa) 4.)The interactive activation and competition (iac) 5.)The auto-associator (aa) 6.)Back-Propagation (bp) and finally 7.)Competitive Learning (cl). Also with the program were two books for further information and background ,these were used for the research. For analysis of conventional computers the Commodore 64 was used using the BASIC programs and other introductory programs.

## METHOD

The study and assessment of neural network computing versus conventional computing was divided into four stages 1.)Research neural networks 2.)Research conventional computers 3.)Simulate neural networks 4.)Simulate conventional computer programs.

Researching neural networks took many hours, and the first week and a half of the research. My background on the subject was very poor and the research was very different. Time was spent in Rome's technical library and reading through material that my mentor Robert Kaminiski supplied me with. After the research on neural networks it was time to simulate programs on the ZENITH DATA SYSTEMS computer. Observations and inferences were made on the networks found in the Explorations In Parallel Distributed Processing software.

The third stage required again research in the library and past knowledge. This step was on conventional computers. Little research was needed though because of the use before. No "hands on" work was performed on the conventional computers at the research site, but rather at a friend's house. Programs included BASIC and PASCAL and little time was spent because ample information already known could make comparisons.

## RESULTS

During the research period results were obtained through use of computer programs and reading of reference manuals. Work with the artificial neural network was noticeably faster and less time consuming. Conventional computers worked in a more drawn out type of manner, that is the work was done but took long periods of execution time as compared to the neural net.

By using PDP programs such as Interactive Activation and Competition (iac) and Constraint Satisfaction (cs) I was able to see the efficiency and bright future of neural networks. Using iac and inputting information with fictitious gangs called the "Jets" and "Sharks" on education, marital status, age, and vocation you could have identified anyone. By changing the weights for a person named "Lance" you could tell whether he was married, single, divorced, whether he was in his 20s, 30's, or 40s, whether he belonged to the Jets or Sharks, and whether he was in high school, junior high, or college. This I found very different and that no conventional computer could do this with only that little input. Using the cs program one could find out which furniture belonged in what room. By changing the weights again for oven and running 100 cycles you could see changes by the stove, drapes, sink, refrigerator, toaster, cupboards, coffeepot, window, walls, ceiling, and small, they all showed to be either 100 or 99 which meant that they all belonged together and described the kitchen.

## CONCLUSION

Artificial neural networks showed to be much better than conventional computers at some tasks. Research in the area of neural networks may be increased due to the fact that they are more efficient than conventional computers. Conventional computers by far are better in accuracy, and always produce the right answer one-hundred percent of the time. But neural networks bring a new field to computers that is the idea of not being always right but being practical like ourselves.

Both networks can commit errors. Computer programmers can enter the wrong information, or damage to hardware and software will result in an error. Probably the biggest problem for the neural networks, as already stated, is that present technology cannot provide answers to how this networks came up with their own answers. This means that if people tend to use neural "nets" they will have become more dependent on computers than ever.

Through this research it is concluded that both systems have great advantages and their own special drawbacks. But the world will continue to use the conventional computer just as much as they might use the neural network system.

## BIBLIOGRAPHY

- Allman, William F. "Designing Computers To Think The Way We Do",  
Technology Review, Vol. 90 (May/June, 1987), 58-65
- Beardsley, Tim. "Neural Networks At Work", Scientific American,  
(November, 1988), 134-135.
- Bien, William. "The Promise of Neural Networks", Science Observer  
(November/December, 1988), 561-564.
- Defense Advanced Research Project Agency (DARPA), Final Report, MIT,  
(October 1987-February 1988), 1 5-26, 11 5-26.
- Enfield, Ronald I. "The Limits Of Software Reliability", Technology Review, Vol. 90, No. 3 (April, 1987) 36-43.
- Jones, William P. and Josiah Hoskins, "Back Propagation", BYTE  
Vol 12, No. 11 (October, 1987) 155-162.
- Lippmann, Richard p. "An Introduction To Computing With Neural Nets",  
IEEE ASSP Magazine, (April, 1987) 4-22.
- Manuel, Tom. "Are Artificial Neural Networks Finally Ready For  
Market?", Electronics, (August, 1988) 85-88.
- Murray, Allan P. and Anthony V.W Smith. "Asynchronous VLSI Neural  
Networks Using Pulse-Stream Arithmetic", IEEE Journal of Solid  
State Circuits, Vol. 23, No. 3, (June, 1988), 688-690.
- O'Reilly, Brian. "Computers That Think Like People", Fortune,  
Vol 119, (February 27, 1989) 90-93.
- Roberts, L. "Are Neural Nets Like The Human Brain?", Science Research  
News, Vol. 243 (January, 1989) 481-482.
- Tank, David W. and John J. Hopfield. "Collective Computation In  
Neuronlike Circuits", Scientific American, Vol. 257, No. 6  
(December, 1987), 104-114.
- Till, John A. "Computer System Architecture", Electronics Design  
(January 12, 1989), 50-64.



## ILLUSTRATIONS

Figure 1-Biological Neuron

Figure 2-Electron Neuron

Figure 3-Weighted Figures

Figure 4-Layered Neurons

Figure 5-Nonlinear Mapping

Figure 6-Representatives Nonlinearties

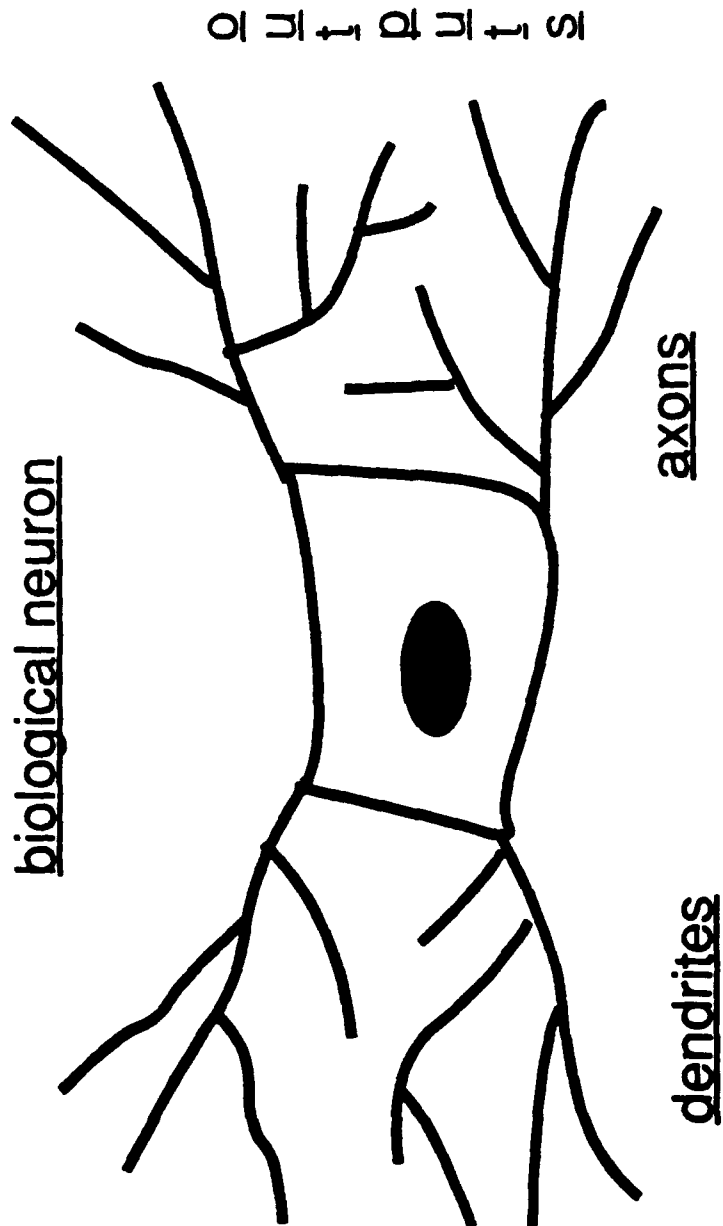


figure 1

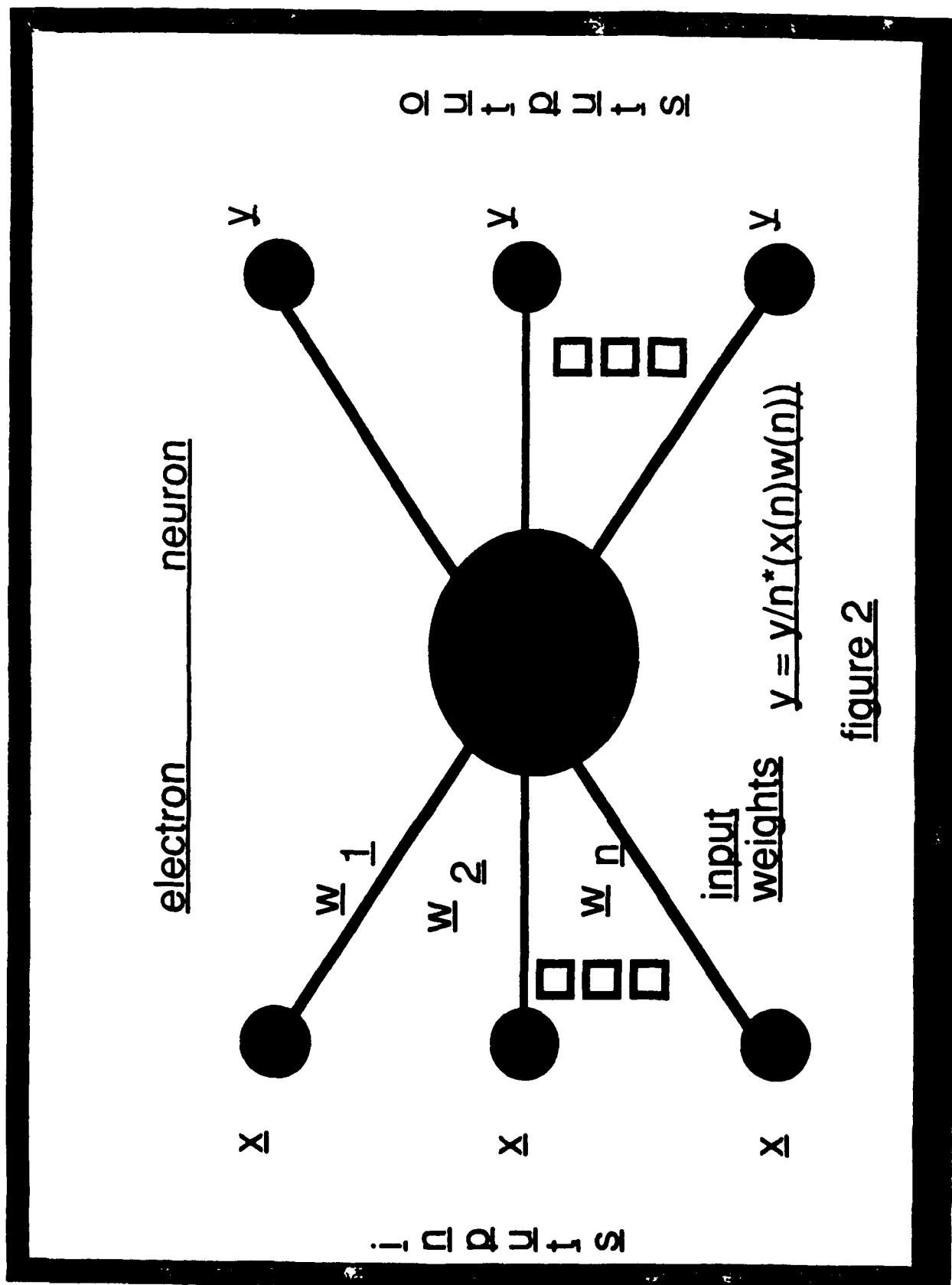
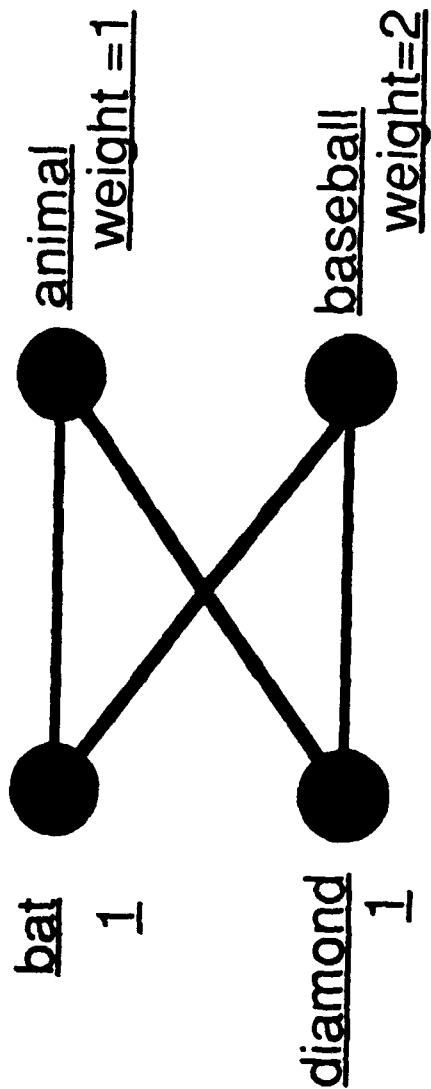


figure 2

i n p u t s

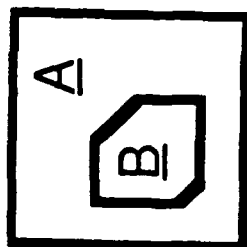
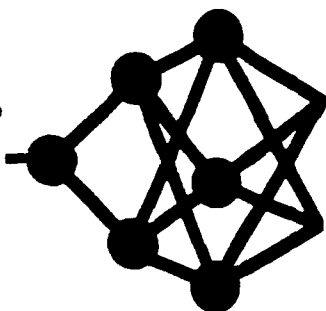


o u t p u t s

Bat is common with animal and baseball, so each has a weight of 1  
Diamond is only common with baseball so the two inputs weigh more towards baseball.

figure 3

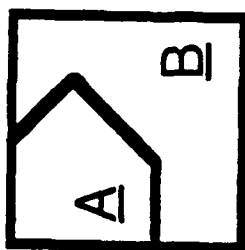
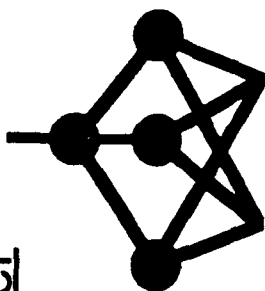
three layer



two  
regions

ARBITRARY

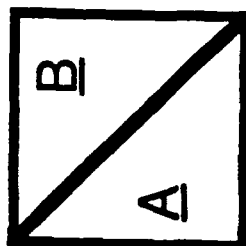
two layer



3 half  
planes

TYPICALLY CONVEX

one layer



1 half  
plane

HALF PLANE

figure 4

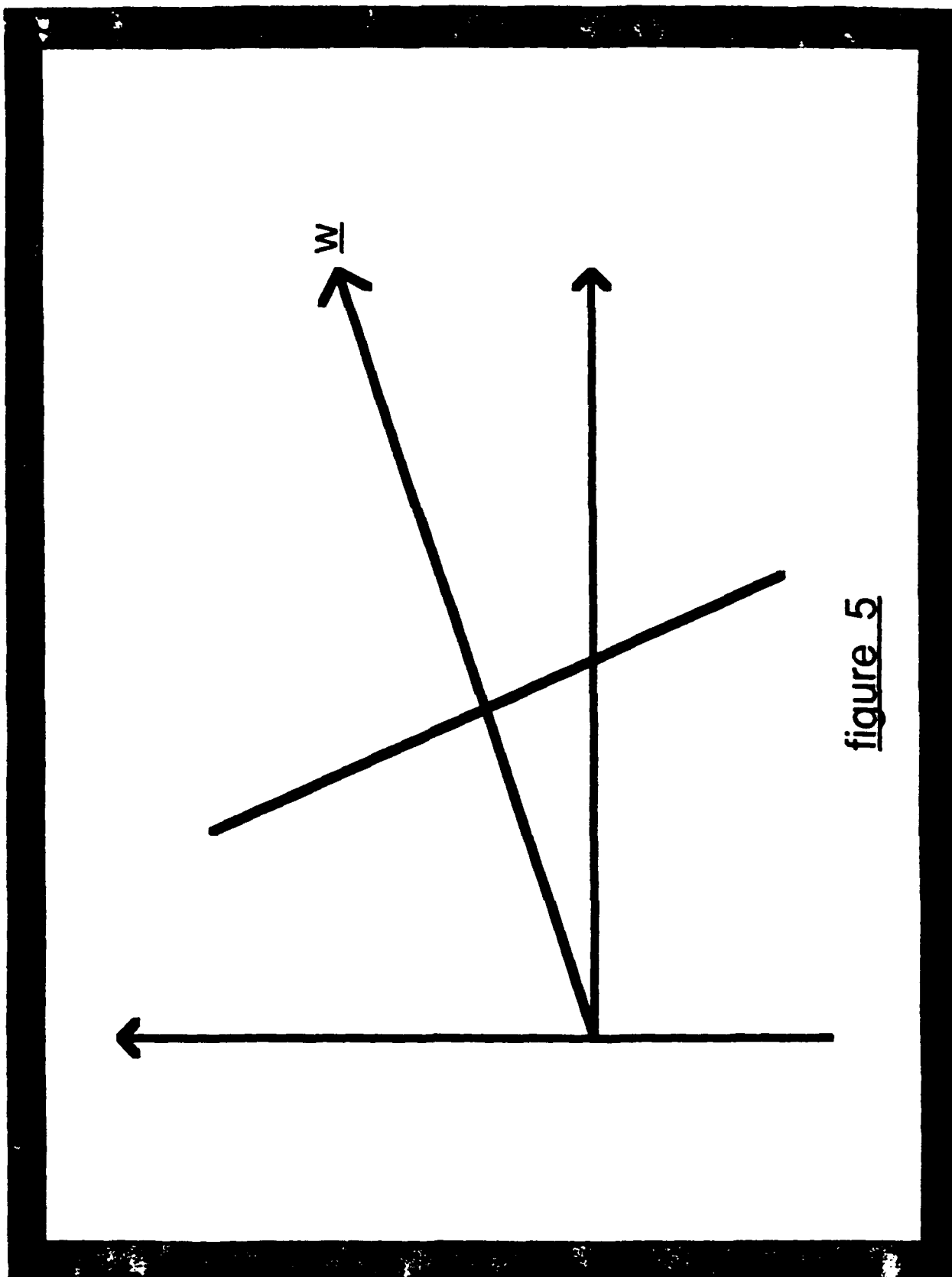
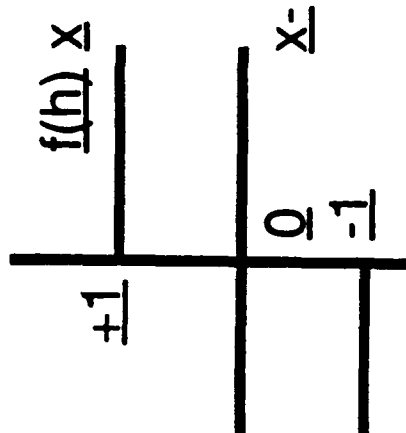
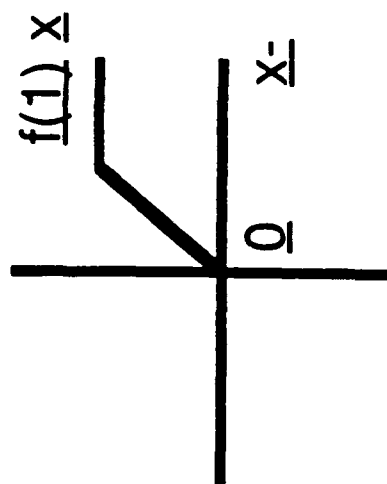


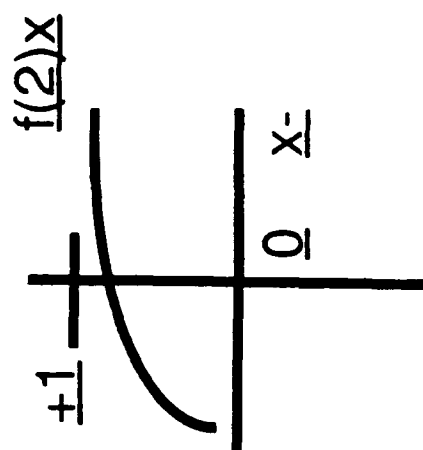
figure 5



hard limiter



threshold logic



sigmoid

Three representative nonlinearities

figure 6

**AFOSR**  
**SUMMER RESEARCH PROGRAM**  
FOR HIGH SCHOOL APPRENTICESHIP 1991

***RDL***

***RESEARCH & DEVELOPMENT LABORATORIES***

***PLACE OF RESEARCH:ROME LABS, GRIFFISS AIR FORCE BASE***

***APPRENTICE:Christopher Moylan***

***LAB FOCAL POINT:Dr Raymond Liuzzi***



## **CORRECTING A PHONE LOG PROGRAM FOR USE WITH MICROSOFT EXCEL**

### **0.1 INTRODUCTION**

My summer work was done at Griffiss Air Force Base in Rome, N.Y. While at Griffiss I was assigned to fix a program for keeping track of commercial long distance phone calls. Since I had never used a Macintosh or Microsoft Excel before I came to Griffiss, and the program I was assigned to fix was written for these two, I had to learn how to use them before I could begin fixing the program. Once I was adept at using these, I began to try to find and eliminate the errors. These errors were causing data to be lost and the program to run very slowly.

### **0.2 ACCOMPLISHMENTS**

During the course of my work I was able to define, correct, and update a number of different errors in the Phone Log Program. These included errors, in the syntax, in referencing, and in the command structure of the program.

### **0.3 SUMMARY OF LONG DISTANCE LOG**

This Phone Log Program along with Excel v2.2 automates the tracking of commercial long distance phone calls. All functions are available under the only menu, PHONE.WHO/WHY, lets you specify who you are

calling and the reason for the call. If you have ever called that person before this routine shows you their phone number for easy dialing. If you need to modify data be sure to click the save box to save changes. Names and reasons in the database can be sorted alphabetically or by frequency of usage. Sorting does not affect the arrangement of names or reasons at this time but the next time you use the program the data will be sorted accordingly. CLASS A EXT. lets you record the extension you are using. When a connection is made click on CALL BEGINS. When the connection is broken click on CALL ENDS. At this time data will be pasted from a temporary buffer to the permanent long distance log. you may re-edit some information at this time if needed before permanent entry. If for some reason you need to cancel a call in progress do it with CANCEL CALL. CANCEL CALL also allows you to remove the last call from the permanent database if necessary. That is the safest way to remove data from your long distance log. Both the temporary data array and the permanent log are protected spreadsheets so you normally can not modify them directly. You can print data at any time using the command PRINT DATA. You may want to delete names, reasons, or actual log data from time to time. To do this you use DELETE DATA. For those occasion when you want to edit some of your log data click EDIT LOG DATA.

#### **0.4 GETTING STARTED**

Since when I started here I had never used a Mac or Excel I had to learn them before I could fix the Phone Log program. I started by going through some Apple tour disks for the Macintosh, these disks taught the basics for the Mac. Then I watched some instructional

videos for Microsoft Excel. These videos, "Getting Started With Microsoft Excel", and "Advanced Techniques for Microsoft Excel", are two very good teaching tools. By having the user make actual spreadsheets as he or she goes through the tapes one can learn much better then by just hearing about it or watching someone else do it. The only problem with the videos is that they did not get very in depth into teaching how to use the macro function. Macros can be used to set up the same spreadsheet over and over, this is what the Phone Log program uses to record data and carry out the commands. On hand were several books on Excel that also turned out to be very useful, the most useful of these books was "Microsoft Excel Functions And Macros". This book defines all the commands in the Excel Command Macro Language and shows how they should be written. This was especially important since only a small error such as the absence of a period or a comma could cause the computer to misunderstand the cell and carry out the wrong command.

## 0.5 FINDING WHAT WAS WRONG

I began correcting errors by first trying the program to see what was wrong. First, the program would not run past the second cell with out registering an error and needing to be told to continue calculating. After a detailed study I determined that syntax errors were a cause of this problem.

## 0.6 ELIMINATION OF SYNTAX ERRORS

Next I wanted to eliminate any other syntax errors that might have

been in the program. This was time consuming because, using the book "Excel Functions And Macros" I had to check each cell to see if the commands were inputted properly. It only takes the smallest of mistakes in the cell to cause an error that can result in major problems with filing data or entering it in the correct format for the Phone Log. Once this was done I had to develop a greater understanding of Excel functions and using custom dialog boxes. This program uses many dialog boxes and I was fairly sure there were some errors in these boxes and their references.

## 0.7 CUSTOM DIALOG BOXES

The book "Excel Functions And Macros" also gives a lot of information on custom dialog boxes and their purposes. Dialog boxes in Excel must have a dialog box definition table somewhere in the macro sheet or another place they can be referenced to easily. For this program the definition tables are in the Phone Data, And Long Distance Log documents that are separate from the macro sheet. That means the macro sheet has to reference to these precise parts of the document in order for the functions to be carried out. The references need be off by only one cell in their coordinates and the program will not be able to extract the right data. This can cause an interruption in the processing and inputting of data. Once I understood this I began to check all the references to dialog box definition tables to see if the coordinates matched those of the corresponding table.

## 0.8 ERRORS IN REFERENCING

I also defined errors in referencing. There was one instance when the

macro sheet referred to the wrong definition table. This caused the program to run out of order and to not record the data into the permanent log. There was also many instances when the macro sheet referred incorrectly to a definition table. These were cases when the wrong coordinates were given and the macros referred to only part of the table or some of the blank space around the table. This was corrected by changing the coordinates to match the location of the correct definition table.

## **0.9 DESCRIPTION OF DIALOG BOX DEFINITION TABLES**

The parts of a dialog box definition table all have their own important purpose. Each definition table must be made up of seven columns. The first column contains numbers which tell the program which of the possible parts and how many to include in the table. These numbers each have an equivalent part such as text or edit boxes. The second and third columns contain horizontal and vertical coordinates. The fourth and fifth columns are for item width and height. Any text to be entered is entered in the sixth column. The seventh column is titled the Initial/Result column, this column has two functions. It can be used to specify the values and settings that appear when the dialog box is open or it can also be used to record values entered in the dialog box when it is open after the dialog box is closed with an OK button.

## **1.0 INCOMPLETE DIALOG DEFINITION TABLES**

If any of these parts are left out the dialog box will not function

properly. In one of the dialog boxes there was no way to tell the computer whether or not you were satisfied with the information. The first column of one of the definition tables was incomplete and left out the OK and CANCEL buttons. This resulted in a stopping of the program because it needed that information to continue. Once I realized that, I inputted the correct numbers into the first column of that dialog boxes definition table. Then the dialog box was complete and that part of the program was working as it should have been.

### **1.1 ILLEGAL REFERENCES**

In several instances there were illegal or incorrect references to dialog box definition tables. These are cases when either the entire table is not referenced or too large an area is referenced and there is blank space included into the table. If the entire table is not referenced the commands outside the referenced area will be ignored and the result will be the same as if the command was not entered. If too large of an area is referenced the the program will stop and tell you that there was an illegal reference. This results in the need to give the command to continue and this slows the program greatly. To fix this problem all that is necessary is to check each reference and be sure that it matches the tables coordinates exactly.

### **1.2 INAPPROPRIATE COMMANDS**

In some instances commands were not appropriate. One such case was when the command DELETE.BAR was followed by the command SHOW.BAR. The command DELETE.BAR is the macro function to delete

a menu bar. The command SHOW.BAR is the function that is used to show a custom menu bar that was added at some previous point in the program. This resulted in an error that stopped the program. I was able to fix this problem with out to much trouble because it was clear that the command ADD.BAR is what should have been in the place of DELETE.BAR. ADD.BAR is the command to add a menu bar according to a previous group of cells which list all the options in that menu.

### 1.3 CONCLUSION

The program had numerous mistakes that were all similar to the ones I mentioned above. It was just a matter of running the program until it registered an error then taking the time to find out why that was an error then to correct it. In the beginning when I first started doing this it was very time consuming because I had such little experience with this type of work. It would take take me a while to isolate the problem and find out why it was an error. Then it would take me an even longer time to find the correction because I had to look everything up to find the proper format, and I still made mistakes. Then towards the end it all came together. I understood why things were wrong and could correct them quickly. This obviously made The work get done a lot faster and also showed me how much I have learned.

### 1.4 FUTURE IMPROVEMENTS

There are still quite a few improvements that could be made to the

long distance log. The program could be better if the dialog box that gives the name, telephone number, address and reason for the call stayed open while the call was in progress. Instead this dialog box closes when you click CALL BEGINS. Also if the place in the Phone Data document where the names addresses and telephone numbers of those who you have called before was more accessible it would be of better use, because you could use it for more than just inputting data into the Long Distance Log. These are some of the improvements that could make this program better that I recommend to someone continuing this project.



Item numbers are numbers in first column dialog box definition table and the dialog box item is the corresponding part of the dialog box.

ITEM NUMBER	DIALOG BOX ITEM
1	Default OK button
2	Cancel button
3	OK button
4	Default CANCEL button
5	Static text
6	Text edit box
7	Integer edit box
8	Number edit box
9	Formula edit box
10	Reference edit box
11	Option button group
12	Option button
13	Check box
14	Group box
15	List box
16	Combination list box
17	Icons

Name	<input type="text" value="Chris Moylan"/>	<input type="button" value="OK"/>
Telephone Number	<input type="text" value="797-6111"/>	
		<input type="button" value="CANCEL"/>
		<input type="checkbox"/> SAVE
City	<input type="text" value="Utica"/>	
State	<input type="text" value="NY"/>	
Company	<input type="text" value="RDL"/>	
Reason For Call	<input type="text" value="Final Report"/>	

DIALOG BOX SHOWING THE NAME, TELEPHONE NUMBER, CITY, STATE, AND COMPANY OF THE PERSON TO BE CALLED USING THE PHONE LOG PROGRAM. THIS DIALOG BOX ALSO SHOWS THE REASON THE CALL IS TO BE MADE

## **Acknowledgements**

I would like to thank the AFOSR and RDL for sponsoring and conducting the High school Apprenticeship Program. I am very grateful to have been given such a great opportunity to grow in my knowledge and understanding of the developing technological field.

I would also like to thank the following people:

Dr. Ray Liuzzi - I thank you for all your instruction and guidance, and for the Coffee Club which helped me make it through the mournings.

Lou Comito-I thank you and so does my football coach for all the donuts and other food you gave me. It was grate to work with you.

Capt. Jeff Grimshaw USAF- thanks for your instruction and for showing a *genuine interest in helping me.*

Craig Anken- thank you for your help and for all the times I needed to get into the lab.

Amy Kudla- thank you for putting up with sharing an office with me and answering my dumb questions.

Debra Panek- thank you for answering my dumb questions and for getting that disk out of the drive for me.

Gina Miller- thank you for all the brownies and cheesecake

1991 USAF-RDL High School Apprenticeship Program

Sponsored by the  
Air Force Office of Scientific Research

Conducted by  
Research and Development Laboratories, Inc.

FINAL REPORT

An Introduction to  
Software Engineering and Parallel Processing

Prepared by: Debra A. Panek  
Research Location: Rome Laboratory, Griffiss Air Force Base  
USAF Mentor: Mr. Joseph Cavano

Date: August 13, 1991

# AN INTRODUCTION TO SOFTWARE ENGINEERING AND PARALLEL PROCESSING

Debra A. Panek

## **Abstract**

The main purpose of my summer was to gain a basic understanding of software engineering, its concepts and environments, and to interact with software engineers and their efforts to advance their field. I have been exposed to new computer systems, their hardware, software and software design, and to high-level languages for coding. I was also introduced to the concepts of parallel computers and transputer communication, and I observed experimentation with their efficiency. Of great personal importance was the opportunity to apply this new technical knowledge and experience to the medical field, where lie my main interests. Thus, I was able to combine computer usage, observation, and literature into a productive summer of research and education.

## **1.0 Introduction**

I spent my summer working at Rome Laboratory on Griffiss Air Force Base in Rome, New York. I was assigned to the Software Engineering Group, where the members test and experiment with new software technologies. The fundamental objectives of software engineering are to provide sound engineering principles, practices, and tools in support of all phases of the software life cycle of planning, development, maintenance, and usage.

## **2.0 Software Engineering**

Throughout the summer, I was exposed to many aspects of software engineering. These include the analysis and complexity of algorithms, software requirements analysis, parallel computing with multiple processors, programming languages, and various computer systems. I was also able to attend numerous presentations on parallel computing from companies, professors, and students.

## **2.1 Algorithms**

An algorithm can be defined as a finite sequence of definite steps that are carried out systematically to solve a problem. It involves fixed rules that interpret stored data as instructions to be executed. So that new complex algorithms need not be continuously developed, basic algorithms are often used in variations and combinations. The time needed to execute an algorithm as a function of input length and the amount of memory space necessary for its execution is known as the algorithm's complexity, which measures the time and/or space needed to execute the algorithm. After the complexity of an algorithm is determined, it can be grouped with others of similar complexity. This classification is part of an algorithm analysis, where problems are compared and characterized to identify their inherent difficulties. Also part of this analysis is a test of optimality; this determines if the algorithm is the best possible to solve the problem. I also studied branch and bound algorithms, which are organized and structured searches of the space of all feasible solutions. This research served as an introduction to the theory, development, and application of algorithms.

## **2.2. Software requirements analysis**

A software requirements analysis is a complete specification of software requirements that is essential for success of the software development effort. There are four specific task effort areas within the analysis: problem recognition, evaluation and synthesis, specification, and review. To satisfy the objectives (i.e., the interface, functions, and design construction) of the analysis, communication between the developer and the requester of the software must be established and maintained.

### **2.2.1. Steps toward problem solving**

A basic order of steps to solve an analysis problem would begin with the specification. A prototype of the software might then be developed, followed by a review by the requester and developer. Finally, modifications are made to the specification until both sides are satisfied. A Fundamental System Model will demonstrate the overall function of the system, and it will always consider the nature of the information. Information flow can be

depicted graphically by a data-flow diagram as data moves from input to output. Refinement of the diagram will almost always be necessary, but any type of information flow (i.e., manual, automatic, or hybrid) can be represented. The diagram emphasizes the flow, not the control of the data. The information continuity must be maintained such that the input and output remain the same to the refinement process.

#### **2.2.2. Data Structures and Representation**

Data structures determine the organization, methods of access, degree of associativity and processing alternatives for information. Some classic data structures are the scalar item, sequential vector, linked list, and hierarchical data structure. A scalar item is the simplest data structure, representing a single element of information. A sequential vector is the most commonly used data structure, and it is a list or continuous group of scalar items. A linked list organizes scalar items, vectors, or spaces into nodes for process as a list, and when multi-linked lists are implemented, it is known as a hierarchical data structure. Hierarchical data structures must be represented unambiguously because of their impact on software design requirements, and can be done so as a hierarchical block diagram or a Warnier diagram. A hierarchical block diagram is set up as a tree structure with multi-level blocks that increase in refinement as more detail is shown. A Warnier diagram is designed as a sideways tree with additional features, such as being able to specify the repetitive nature of a category or quantity.

#### **2.2.3. Database Requirements**

A database is a collection of information organized so that it allows and promotes access, analysis, and reporting. The logical architecture of a database is defined by a schema that represents the definitions of the relationships among the data elements. After the overall objectives and range of the system for which the database is to be developed are understood, a complete and specified information model is developed.

#### **2.2.4. Software Requirements Specification**

The software requirements specification is the part of the requirements analysis that is delivered to the requester. It extends the range of the system, the scope, by establishing a complete information description, an explicit functional description, suitable validation standards, and other pertinent data for the requirements. The specification can be set up as an outline, beginning with an introduction that states its goals, the objectives of the software. The information description will detail the problem that the software must solve, documenting both the information flow and the structure. This may include data-flow diagrams, a data dictionary, and internal interfaces. The functional description describes the procedural fine points for each function that is required to solve the problem, which also consists of a processing narrative for each function, and stated and justified design constraints. The validation criteria section is the absolute review of the information and functional requirements. These requirements are often composed of performance bounds, classes of tests, an expected software response, and other individualized special considerations. Finally, the software requirements specification concludes with a bibliography referencing documents relating to the software, and an appendix for supplementation.

#### **2.3 Parallel computing and multiple processors**

The main concentration of the software engineers with whom I worked was in the area of parallel processing. This is an alternative to sequential computation, where the computer can perform only one operation at a time consecutively. With multiple processors, numerous processors may work on pieces of a problem simultaneously, thus reducing absolute computational complexity and the problem running time(increasing its speed). The quality, speed, and clarity of the communication is key, for if the communication is faulty or tie-ups occur because of deadlocked resources, parallel computing will not demonstrate substantial speed benefits. As new software technologies are developed, parallel processing will take on a greater role in software engineering.



## **2.4        Programming Language Experiences**

Before this summer, I had programming experience only in BASIC, with some exposure to PASCAL. While working at Rome Laboratory, I was introduced to various high-level languages in reading materials, and began simple programming in C and FORTRAN. I was able to aid a computer scientist in my group, Mr. Rick Metzger, by combining two FORTRAN programs into one initialization program that created a candidate and a target file for a data fusion project. Thus, through both research and implementation, I experienced the difficulties involved with high-level programming that often evolve from the limitations of the computer software, hardware, and architecture.

## **2.5    Computer systems, hardware, and software**

My first task and subsequent accomplishment at Rome Laboratory was to become familiar with the Macintosh computer and its applications. In addition to the Macintosh, I was able to learn about and have access to a Sun Microsystems computer and its SparcStation. It was on the Sun that the capabilities and intentions of parallel computing were explained to me. I observed Mr. Metzger and visiting professor Dr. John Antonio work on their data fusion program, and also watched summer aide Amy Kudla adjust and improve an animation tool, both which made use of transputer boards within the Sun. I worked through various introductory tutorials and experimented with different software, including word processors, graphics, media-makers, and filing systems. I also studied information about the differences between compilers and interpreters for computer languages. Compilers are software that translate high-level language programs into low-level machine code that can be executed independently, while interpreters execute programs one statement at a time, transferring each high-level construct into machine instructions immediately. Most high-level languages use compilers because of their speed advantage and greater convenience for program development.

## **2.6        Presentations**

While an apprentice at Rome Laboratory, I attended numerous presentations on parallel processing offered by visiting professors, graduate students, and

companies. The topics included parallel architectures, debugging and optimizing programs for parallel computers, and rapid prototyping of real-time systems. Though the content of the workshops was usually beyond my immediate comprehension, I was often able to relate the material to new information I was just beginning to grasp. The presentations also offered me a taste of the innovative work currently being done in the field of parallel processing.

### **3.0 Results**

I consider the results of my summer in the High School Apprenticeship Program to be overwhelmingly positive, especially my personal gains. I now have a greater knowledge of software engineering, computer science, and the details and requirements of the occupations. From exposure to new computer systems and their hardware and software, I gained valuable experience that will become a strong base for my future courses and career interests. I was also given the opportunity to interact with professional engineers and scientists in their environment, absorbing and learning from their technical expertise and experience in the 'real world' work force.

### **4.0 Conclusion**

As my summer at Rome Laboratory at Griffiss Air Force Base ends, I am grateful for having had the opportunities allotted to me. I was able to receive an additional education before beginning formal studies in a college setting, thus honing my technical skills in areas where growth was necessary. Such extensive contact with the field of software engineering has left me much more appreciative and impressed by the accomplishments and potential of the field, and I am certain this background will greatly influence my future studies and career plans.

## Acknowledgements

I would like to thank the AFOSR and RDL for sponsoring and conducting the High School Apprenticeship Program. I am very grateful for the opportunity to become involved and more experienced in such a scientific and technical field, and I am sure my research here will be of great benefit to me in my future studies. I wish also to express my appreciation to the following people:

Mr. Joseph Cavano- for all your instruction, guidance, and support, for making me try and smile when my mouth couldn't, for becoming a friend besides a mentor- thank you.

Rick Metzger- for pushing me to work harder by example of your astonishing work ethic, for making me feel immediately welcome- thank you. Your stamina in everything is amazing, but 6 kids? - Good Luck!!

Milissa Benincasa- for demonstrating a woman can really hold her own and succeed in a male-dominated environment, for getting me into and helping me through the summer- thank you!

Dr. John Antonio- for treating me only as someone trying to learn and helping me to do so, for acting as a friend and not just a professor- thank you (even though you'll probably never see this!!).

Amy Kudla- for helping me learn my way around, and for being nice even when I was a nuisance- thank you. Good luck with everything!

Chris Moylan- for taking walks and sharing complaints with me- thank you. You'll have to find someone new for daily target practice, though! Good luck getting to and through Annapolis!

## References

- Pressman, Roger S. Software Engineering: A Practitioner's Approach.  
McGraw-Hill Book Co., Inc., New York. 1982.
- Vick, C.R., and Ramamoorthy, C.V., eds. Handbook of Software  
Engineering. Van Nostrand Reinhold Co., Inc., New York: 1984.
- Tesler, Lawrence G. "Programming Languages", Scientific American. Vol.  
251, September 1984.
- Kay, Alan. "Computer Software", Scientific American. Vol. 251, September 1984.
- Wirth, Niklaus. "Data Structures and Algorithms", Scientific American.  
Vol. 251, September 1984.

## DATABASE MANIPULATION APPLICATION FOR USE WITH PARADOX

Thatcher J. Pospiech

Over the summer I spent my apprenticeship at Rome Laboratory at Griffiss AFB and was assigned to a project called "Database Manipulation Application for use with Paradox". On this project I came in contact with numerous different programs, several different programming languages, and multiple computer devices.

The final goal of the project was to develop an application that would allow data to be downloaded from a from a centralized database VAX computer and imported into an environment on a PC in which it could be manipulated. The application had to allow any user, even one unfamiliar in the environment used, to easily manipulate and add data to the records downloaded.

The problem had arisen involving the use of remote dumb terminals for accessing Oracle as the main database system for data stored for each directorate. Each directorate or section needs the same data entered into the Oracle database, but then each directorate needs its own data and records. Also, directorates needs more fields than Oracle provides, since each directorate has its own unique fields that must be used in conjunction with the primary database.

There were two possible ways of solving this problem. One, the maintainers of the primary database could enter new fields into Oracle for each separate directorate but this would result in a huge number of fields that were useless to all the other directorates. Also, it would make the database even larger than it was, reducing speed of processing and manipulation of the data. Two, the directorates that required their own data could download the information into an environment on a PC. Once in this environment an application program would have to be developed to manipulate the data the desired amount and simple enough to facilitate use. The second possibility was chosen as it would allow easier manipulation of the data. Hence, the creation of this project in order to develop such an application.

Paradox 3.5 was chosen as the environment that data would be downloaded into because of its relative ease of use and because an application was already up and running in Paradox. Paradox's unique program language, PAL (Paradox Application Language) was used to build the application.

Several problems arose during the course of the project. Each of these had to be solved in turn. One of the first was how to get the data on the MISVAX in Oracle from the mainframe system and into a PC and eventually into the Paradox environment we were designing.

The first step was establishing an account on the MISVAX system to be able to access Oracle. Then a file originating

from Oracle that contained the needed data had to be collected from the MISVAX. This was accomplished using an SQR, queries from a PC interfaced to the mainframe using a network interface unit(NIU), and Kermit. First, an SQR program needed to be written to abstract the data from Oracle in an altered delimited form that Paradox was set up to accept. This SQR program was then executed on the MISVAX to generate the track data file. The next step was downloading the track data file from the MISVAX. A PC was set up with an NIU attached to it. Then a protocol language, Kermit, was used to attach to the Misvax. Using Kermit the track data file was downloaded from the MISVAX onto the PC.

At this point the file could be imported as delimited ASCII text into Paradox. As the data was imported, Paradox separated the fields into columns and labeled the fields Field-1, Field-2, Field-3, and etc up to Field-32. Then each field was translated to its real name and the table was saved. Using this process a user is now able to empty this table and update it whenever they download the track data file from the MISVAX.

To begin this project I had to learn how to use Paradox 3.5. This is an extremely powerful relational database program that has its own programming language PAL(Paradox Application Language). PAL is an high level programming language much like PASCAL but it is designed specifically for Paradox. As with all programming languages PAL is complex and

a problem can take a large amount of time to diagnose.

There were several major sections to the application. The two that I worked on were the "Data" and "Report" portions. Each of these consisted of several different scripts connected by numerous menus.

The "Data" section was the first portion I worked on. Its function was to allow the user to edit, add, and delete records. The Data script itself was just the menu for that section, it only controlled which scripts would be called up next. The data script branched into three different scripts: one for each of the menu choices; Add Records, Edit Records, and Delete Records.

The Add Records script served the function of allowing the user to add records to either the users or systems tables. The systems table contained the records entered on the MISVAX in Oracle, these records were password protected to prevent unauthorized personnel from tampering with them. The user table is where the individual directorates enter their own data for the records. When the user wants to add records to the system's table the program allows him to fill in the information on a pre-fabricated form and then appends the new record to the end of the systems table. If the user wants to add user records, the computer allows the user to view the systems table and select the record they wish to append.

The Edit Records script allows the user to edit and change the contents of the current records that are in the



file. As with the Add Records script the systems table in this script is also password protected to prevent unauthorized tampering or accidental loss of crucial data. The authorized user will be allowed complete freedom of movement to alter any field in any record they wish except the job order number. Those are protected from editing by anyone because they are crucial in identifying the individual records.

Delete Records script allows the user to move through the table and choose a record to delete. It then puts that record into a pre-fabricated form so that the user can delete the entire record or only selected fields in that record. The script will ask the user several times if they want to delete the record so that there are no mistaken deletions. As in the Add and Edit scripts, the systems records are password protected.

The reports section is the group of scripts that will allow the user to generate reports of the data from the users and systems tables. The user has three general categories of report generators that they can choose from. They can choose from several standard reports, go through a step by step help process to generate their own custom reports, or just go and make their own reports with no help other than that which is already embedded in Paradox.

The standard reports choice will bring the user to a menu where they will have a choice from several different reports. One standard report generates information concerning the

funding amount, fund source, fund user and other information concerning the funding of a project. A second one gives information concerning the contract a project is under. A third will allow the user to isolate individual records by their job order numbers and print out selected information from each one. The fourth standard report provides a list of engineers and other personnel working on each project and other related information.

I will never get to see the final results of the project, because of the massive size and complexity of the application. We could not finish all the scripts needed to make the application run as requested. Those parts that have been finished are in working order and will be put into use when the application is finally able to run fully. One thing we discovered while building the application is that because of its size, many computers will not be able to have anything, other than that application running.

I feel that my experience in the high school apprenticeship program will prove very valuable. I learned many things in the eight weeks I was working. The apprenticeship program has given me valuable experience that I feel I will find useful later on. It has taught me responsibility, how to work well with others, management of available time and resources, and the gratification of solving a problem. The program has also strengthened my interest in the computer science and engineering fields, in which I hope

to pursue a career. Also I discovered that I would like to obtain a position with work and atmosphere similar to the ones I had this summer.

## **SECOND-GENERATION SOCC**

**Jason M. Riordan**

### **Abstract**

This paper presents an general overview of the second-generation SOCC. Topics given special attention are the software configuration and internal and external SOCC communications. The possibility of implementing FDDI as an internal data network for the SOCC is examined and current FDDI is found to be inadequate for this purpose. Future applications of this technology are discussed, and its application to multi-media integration should create a more secure environment.

### **1. Introduction**

#### **1.1. Second-Generation SOCC**

The second-generation SOCC, currently in development, is a replacement for the aging Sector Operations Control Center software, using, for the most part, off-the-shelf commercial hardware and government maintained software. Although the existing system is extremely reliable and efficient, a second-generation system is needed to combine SOCC information with other relevant data from new sensors and display them in the clearest, most concise way. Information is the current SOCC is not easily accessible and is, except for the track display, text-oriented.

The goal of the second-generation SOCC is to speed up the process of detecting hostile or potentially dangerous aircraft. This can be accomplished by the integration of graphical data displays of various representations, including tracks, weather, flight plans, and cartographical features.

The various programs that comprise the SOCC software utilize distributed

processing. They are organized into stations that perform a logical set of functions, including top-level display, intermediate translation, and external communications.

Internal communication between stations is currently performed via Ethernet. Recently, a new fiber-optic networking technology called FDDI has become popular among larger networks. The feasibility of using this technology and promised future advancements in this field is considered for the SOCC environment.

External communication between SOCCs involves a wide variety of software protocols to deal with the unique transmission methods employed by the SOCC. The second-generation SOCC remains compatible with existing external communication systems. Protocols and software arrangements for the purpose are discussed.

## **2. Second-Generation SOCC Overview**

Air defense for the United States and Canada is accomplished jointly by the two countries, in a joint system of command and control known as the North American Air Defense (NORAD). The Continental United States uses four Sector Operations Control Centers, or SOCCs, to handle the functions for their area. The Canadians accomplish Command and Control through two Regional Operations Control centers, and Alaska has a single SOCC.

The SOCC's main purposes include air surveillance, identification, and battle management. After entering into the North American Air Defense Identification Zone (ADIZ), all aircraft must be identified. Detection of the aircraft crossing the SOCC and/or ADIZ boundaries is accomplished by radar sites scattered throughout the SOCC region. Data on detections is sent to the SOCC, where it is combined with data from other sites to produce an air picture of the entire sector. The SOCC also serves as a command and control site for neutralization of any air threat. The battle staff will formulate the mission plan and convenes in the section of the SOCC known as the Battle Cab. Their responsibility is to accomplish the long range planning and

direction to assure a protective Air Defense.

The second-generation SOCC software is designed as a distributed processing system. Each station can be classified into one of three categories: air identification, battle management, or communications. The air identification and battle management categories correspond to the functions mentioned above. Communication involves both internal and external data and voice connections. The SOCC receives information from the radar and other sensor sites, weather networks, the Federal Aviation Administration, Ministry of Transport, other SOCC's, and the Regional Operations Command Control Center via voice and data lines.

Aircraft entering the ADIZ are detected by radar sites scattered over North America. Radar data is sent to the SOCC, where it is combined with data from other sites to produce an air picture of the entire sector. Air operations personnel are responsible to insure all aircraft entering the ADIZ are identified as friendly or hostile. Interceptor missions are directed against the latter. The Battle Cab portion of the SOCC provides battle management. This group is not concerned with detecting hostile targets, but to insure that resources are available for the elimination of the threat.

The second-generation SOCC is designed around a distributed system architecture connected via Ethernet. It is divided into three sections -- air identification, battle management, and communications. Each section has a certain level in the distributed architecture that it falls into, covering the spectrum from low-level data processing to the top-level graphic interface.

Air identification is controlled by the top-level Radar Analysis Program. This program is subdivided into three modules: the data interpreter, command processor, and operator display. The data interpreter handles the translation of track data into displayable information, which can be shown in graphical format. This is accomplished by the operator display, which is in turn controlled by the command

processor.

The battle management display involves another top-level program, the Battle Cab Air Defense Display. This is functionally similar to the Radar Analysis Program, but all three modules are combined into a single program module.

The communications section contains most of the common and controls the transmission of data from one point to another or the translation of that data into a useable format. There are four main sources for data that the second-generation SOCC uses:

The most important is live radar data. This information is entered the SOCC by modem on a dedicated line. Using a combination of hardware and software, the lines are decoded and produce up to 24 synchronous data streams. The resulting data is passed on to a Server program. The Server checks the validity of the radar messages and transmits them to each station on the SOCC network.

The decoded radar data can be received directly by the Radar Analysis Program, a tracker, or a record/playback utility. Raw radar data received by the Radar Analysis Program is displayed as a graphical interpretation of the data packet. Data that is sent to the tracker is interpreted, using an algorithmic method, into aircraft detections, or tracks. The tracker transmits the resulting tracks to the Radar Analysis Program or Battle Cab Air Defense Display for graphical interpretation and display. The record/playback utility allows live radar data to be saved in binary format so that it can be replayed at a later time. The prerecorded data can be output in its original format in real time, allowing an incident to be shown exactly as it happened. The utility can record and playback raw, track, or both types of radar data.

The second-generation SOCC also relies on live weather data from the Air Weather Network. Weather reports are received in ASCII format from stations throughout the nation. The information is interpreted for points between these

stations by an intermediate-level program, whose output is broadcasted on the network. The resulting data is received by the Radar Analysis Program and Battle Cab Air Defense Display and is displayed in graphical format.

The raw ASCII weather information is also used to modify an information file. Any weather station can be queried from either of the two top-level programs, and the information from this file will be displayed.

Also, flight plan data is received from the Federal Aviation Administration. This information includes the planned courses for commercial aircraft entering U.S. airspace. It is received in ASCII format, but is passed through an interpreter to form a graphical representation. The Radar Analysis Program will display this so that commercial flights can be identified more easily. Information on these flights can also be displayed in a text format to show the planned course in greater detail.

Since there are seven SOCCs that comprise the North American Air Defense System, they must have some means of communication between them. This is accomplished by lateral tell messages, which provide binary communication between two SOCCs via a dedicated line. The internal track data structures are compatible with the lateral tell message structure, allowing them to be sent or received with no format translation. The messages can be integrated and displayed transparently.

### **3. Basic ANSI FDDI Standard**

The American National Standards Institute defined FDDI with adherence to the Open System Interconnection (OSI) Reference Model. OSI conformity allowed ANSI to implement adaptations of the standard while retaining compatibility with the original. Examples of such cases include the use of alternative fibers and the incorporation of circuit-switched services.

The physical FDDI network is organized into dual counter-rotating rings, which are designated the primary and secondary. Under normal operation, these act



as separate single token rings. When a fault is encountered, however, the station preceding the fault will redirect its primary output to its secondary output. Ultimately, the data will reach the station on the other side of the fault, which will also redirect its primary output to its secondary output, thereby isolating the fault. This will insure continued network operation, even when a main link is damaged.

FDDI supports a hierarchic structured network. This is accomplished through the use of a concentrator, which redirects its input to the input of a lower station. The lower station's output is mapped onto the input of another lower station, continuing on until the last lower station is reached. The concentrator then maps the last lower station's output onto its own. Concentrators can be nested, allowing an infinite depth of stations with access to the same data, and can have any number of ports for lower stations. If there is a fault between a concentrator and a station, the concentrator will pass the information on to the next station, allowing isolation of any lower stations that are disabled.

Stations and concentrators can be either single or dual attachment. All stations and concentrators in the trunk ring are dual attachment. That is, they connect to both the primary and secondary rings and provide for uninterrupted network operation. Lower stations are all single attachment and connect to the primary ring only. Any lower station that is not active will not affect the network operation in any way (it can be disabled at the concentrator), while continuous operation at the trunk level is assured.

The unenhanced FDDI standard is defined in the following layered standards, each listed with its respective ISO location:

1. Physical Medium Dependant (PMD) - lower Physical Layer
2. Physical Layer Protocol (PHY) - upper Physical Layer
3. Medium Access Control (MAC) - lower Data Link Layer
4. Station Management (SMT) - Physical Layer, Data Link Layer

The basic PMD layer requires the use of 62.5/125 multi-mode fiber and 1300nm transmitters. This will allow for inter-station connections of up to 2km and transfer rates of 100Mbit/s.

The PHY layer defines a 4 of 5 coding scheme. Each possible nibble, or 4 bits, of data is sent as one of 32 possible 5 bit symbols. The remaining 16 symbols are dedicated to control functions. This method produces an 80% efficiency and requires a Baud rate of 125 MHz.

FDDI implements a distributed clocking architecture. The data transfer rate for FDDI is too high for centralized clocking; clock jitter would accumulate at each successive station. PHY requires that information be clocked in to a FIFO buffer, known as the elasticity buffer, at the transmitting station's clock rate. The information is clocked out to the next station at the station's own clock rate. Smoothing functions are incorporated into the PHY to compensate for the rate differences. These functions will regulate the distribution of IDLE messages to prevent the destruction of a frame due to their removal (a minimum of 12 IDLE messages is necessary between packets).

The MAC layer controls access to the ring with the use of a token passed among the stations on the network. A station may transmit data once the token is captured. After transmission, the token is passed on to the next station. Stations that are not transmitting merely repeat the information received. These stations also compare the destination address in the incoming frames to their own address. If the addresses are identical, the frame is copied to local storage and the supervising process is alerted.

Basic FDDI supports two types of transmission -- asynchronous and synchronous. Bandwidth for synchronous transmission is usually preallocated and the response time is predictable. Synchronous transmission is allowed whenever a station captures a token and the full preallocated bandwidth is at the station's

disposal. The station does not have to use all of its allocated bandwidth.

Asynchronous transmission is controlled by a Token Rotation Timer (TRT). During ring initialization, a universal Target Token Rotation Time is established. When a token is received, the Token Rotation Timer is copied and then reset. If the value of the TRT exceeded the TTRT, no asynchronous transmission is permitted. Otherwise, asynchronous bandwidth is allocated from the bandwidth not used by synchronous transfer. This arrangement limits the TRT to twice the TTRT and the average TRT to no more than TTRT.

The Station Management standard, or SMT, occupies both the first and second OSI layers, and controls the overall operation of the FDDI ring. SMT provides for detection of failures and the ability to bypass such failures by reconfiguration of the ring. SMT also creates point-to-point links by using primitive signaling techniques.

#### **4. Enhancements To FDDI**

Since the original FDDI, many adaptations of the original standards have been introduced. The first was an alternative version of PMD, the SMF-PMD, that facilitates the use of single mode fiber and laser diodes. SMF-PMD permits the distance between stations to be in the area of 60km.

Although in the very early stages of development, the SONET Physical layer Mapping standard, or SPM, is an alternate PMD allowing connection to either end of SONET. SPM specifies the conversion from FDDI signal to the SONET STS-3c envelope.

There are also adaptations that facilitate circuit-switched transfers alongside FDDI's packet-switched transfers. Commonly referred to as FDDI-II, this group of standards will allow services such as telephony to coexist on the same network as data transfers. The Hybrid Ring Control standard (HRC) occupies the lower portion of the Data Link Layer, and defines an isynchronous version of MAC (I-MAC) and a Hybrid Multiplexer (H-MUX).

FDDI-II divides the available bandwidth into sixteen Wide-Band Channel units of 6.144 Mbit/s. These may be dedicated to either basic FDDI packet transfer or to isynchronous transmission. Each WBC is allocated independently and can be further allocated by a station into smaller units of 8 Kbit/s or multiples thereof.

## **5. SOCC Implementations Of FDDI-Based Networks**

The first problem evident with a theoretical implementation of basic FDDI, or any token-passing network in a SOCC environment, is its single-station addressing. An FDDI frame can be addressed to one, and only one, station at any one time. This would yield a problem with all the SOCC Communication modules, such as the Server. There would either have to be a dedicated Server, Tracker, and record/playback utility for each top-level program, or the Server and Tracker would have to address each top-level program separately.

Dedicating a Server, tracker, and record/playback utility to each workstation would require four times the stations in the original configuration. Every transmission would circumnavigate the network back to its originating station before it can be removed. The amount of traffic on the network, therefore, could increase by three times the original for each station added (the tracker and playback utility never transmit at the same time). The total transmission could be as large as 150 times the original Server transmissions with 50 top-level programs executing.

Addressing each station separately will eliminate the first problem and lessen the second. However, an extremely fast Server would be required and 50 times the original traffic rate would still be expected. It is clear that if a fiber-optic network is implemented in the SOCC, it would require multiple addressing.

Also, even in a non-standard star configuration, a fiber network would not dramatically increase SOCC communications. Theoretical data transfer rates in FDDI should reach 100 Mbit/s, as compared to 10Mbit/s for the existing Ethernet. The data

rates needed should also, in theory, not average more than 20Kbit/s per top-level program (assuming a realistic number of stations). At this rate, a significant improvement would not be evident.

The difference that could exist between theoretical and observed performance can be accounted for by the overhead, especially the protocol. This overhead will limit the transfer rate to a value that is, for the most part, not related to the medium type.

These facts imply that FDDI would not dramatically increase network performance in a SOCC environment. However, future applications of fiber communications that are in development may cause a greater influence.

An issue of prime importance to a SOCC is security. This may extend to documents and telephone contacts as well as battle plans. Using a dual mode network, such as FDDI-II, documents can be electronically stored in on-line computers and telephones can be made impossible to tap. The documents would be digitized images stored on a high-volume medium. When requested, the document would be sent via the network to the requisite. Sufficient bandwidth could be allocated to provide real-time or near real-time transfer rate, dependant on network traffic. Since detection of a low-level signal current can be employed at all stations, security breaches would be impossible. Using an encrypting transmitter and decrypting receiver at stations eligible to receive the document would prevent unauthorized requests. Satellite photos and live video could also be transmitted in this manner.

Telephony can be done in the same way. Up to 48 T1 lines can be accessed simultaneously at 1.544 Mbits/s each while reserving 3 WBCs for other data transfers. (if the video were employed, more WBCs would need to be reserved). This rate would be sufficient for all internal calls and intercom connections, and security reasons would be satisfied.

In this configuration, a main ring would contain a Server, Tracker, branches

for telephone and/or video service (and servers), and a branch connecting into a star network that attaches each station running a top-level program. This would supply additional stations attached to the ring, or stations attached to ring concentrators, with the ability to request the server for data, documents, or use the integrated telephone service.

To insure security, the network should be closed except for secure connections to other military facilities.

## **6. Inter-SOCC Communication Systems**

Communications between SOCCs is performed via modem over dedicated lines at a transfer rate of 2400 baud. They are synchronous message transfers that are separated by IDLE messages. Each byte of each message is separated by a mark bit, so 9 bits must be read to decipher a byte.

Inter-SOCC transfers are implemented so that the lateral tell messages can be received in the same manner and on the same station as radar information. This arrangement will allow a SOCC to examine radar detections outside its sector, as well as simplify communications for SOCC personnel.

The ideal hardware for implementing this system is Rades, a radar communications system developed by the Air Force. It involves a microprocessor dedicated to handling communications for 8 channels. Each one is located on an expansion card, and can address the same area of memory as its host. Also, there may be more than one card installed at one time. The system can currently be used for transmission of a maximum of 24 radar data channels.

Since memory is addressed by multiple processors, the host may download executable code to each communications microprocessor. This multi-processing environment is easily managed by the host. The environment can be modified by substituting downloadable software for protocol changes and from within a common

environment for non-protocol related tasks.

## **7. Conclusion**

The main purpose of the second-generation SOCC is to allow for easier identification of aircraft that may have hostile intentions. Its distributed architecture allows full integration of data from various sources. The information that was given previously shows only the basic modules -- some of lesser importance have not been examined or not examined to their fullest extent. The second-generation SOCC will achieve its purpose, allowing greater time for reactions to and preparation for aircraft that threaten the nation.

At the present time, FDDI implementation as SOCC data network would have little or even negative impact upon the second-generation SOCC. It would not improve the throughput rate by a large margin, and would negatively impact the overall data amount. A enhanced dual mode network, however, could provide a more secure and convenient document retrieval and internal voice communications system.

**AFOSR High School Apprenticeship Program**

**July 22-August 16 1991**

**Gene Salerno**



### Acknowledgements

I must first thank my mentor, Mr. Richard Smith. He took the time to answer any question and explain why, not just what. His effort at providing me with challenging problems made my stay an enjoyable one.

And thanks to Lt. J. Scott Goldstein for providing me with his sense of humor as well as his computer expertise.

Thank you UES for the invaluable experience and an excellent opportunity.

## Filter/Graphing Program

Much of my time at Rome Labs was spent writing programs for Mr. Smith's use. The following program is an example of my work. I was asked to use the C programming language to write a program that would create wave-shaping filters. Given an input wavelet and a desired wavelet, the program uses a method of convolutions, as described by R. F. Mereu, to create a series of filters, which evolve with every iteration. Each new filter produces a cleaner output than the one before. The program also plots the inputs, filters, and output.

---

### Filter Design

A=Given Wavelet    D=Desired Wavelet    F=Filter

\*=Convolution    A\*F=D

F0=Wavelet A reversed in time

$F = (F1 * F2 * F3 * \dots * Fn) * (F0 * D)$

F1=(A\*F0) With signs of alternate terms changed

F2=(A\*F0\*F1) With signs of alternate non-zero terms changed

F3=(A\*F0\*F1\*F2)                "                "

|

F<sub>n</sub>=(A\*F0\*F1\*F2\*F3. . .F<sub>n-1</sub>)                "                "

n=Number of sub-filters

---

```

/*****
/*
/* THIS PROGRAM USES THE R. F. MERUE WAVE-SHAPING FILTER PROCESS TO
/* COMPUTE A SERIES OF EVOLVING FILTERS WHICH EVENTUALLY PRODUCE A DESIRED
/* WAVELET, HAVING BEEN GIVEN THE VALUES FOR AN INPUT WAVELET (THE ONE YOU
/* HAVE) AND A DESIRED WAVELET (THE ONE YOU WANT).
/*
/*
/*****

#include <GRAPHICS.H>
#include <CONIO.H>
#include <MATH.H>

/*****
/*
/* <<<<STRUCTURE DEFINITION FOR A SERIES AND CORRESPONDING INFORMATION>>>>
/* FIELD ONE IS AN ARRAY OF UP TO 600 VALUES (A SERIES OF 600 TIME SAMPLES)
/* FIELD TWO IS AN INTEGER VALUE DENOTING THE CURRENT LENGTH OF THE SERIES
/* FIELD THREE IS A FLOATING POINT VARIABLE HOLDING THE LARGEST VALUE OF
/* THE SERIES, THIS IS USED FOR SCALING
/*
/*
/*****

typedef struct
{
    float val[600];
    int len;
    float big;
}series;

series wavebase, filtbases, subfilters, temp, desired, wavelet;

/*****
/*
/* THE WAVEBASE IS THE COLLECTION OF THE GIVEN WAVELET CONVOLVED WITH
/* ITSELF REVERSED IN TIME AND THEN WITH THE RESULTING SUBFILTERS PRODUCED.
/* THE FILTBASE IS THE CONVOLUTION OF THE GIVEN WAVELET REVERSED IN TIME
/* WITH THE DESIRED WAVELET.
/* THE SUBFILTERS IS THE COLLECTION OF CONVOLVED SUBFILTERS AS PRODUCED
/* USING THE WAVEBASE.
/*
/*
/*****

/*****
/* FUNCTION DECLARATION SECTION
/*****

void startup();
void putin(series *newsr, series oldsr);
void makefilter();
void alternate(series oldbase, series *newbase);
void convolve(series moving, series stationary, series *store);
void graph(series wave, int grtype);
void startgraph(series wave, int grtype);
void tc();
void numfilters();
int cont();

/*****
/*
/* FUNCTION WILL CONVOLVE TWO SERIES AND PLACE THE RESULTANT SERIES INTO A
/* THIRD VARIABLE OF TYPE SERIES
/*
/*
/*****

```

```

VOID CONVOLVE(SERIES MOVING, SERIES STATIONARY, SERIES *STORE)

{INT STAYCOUNT, MOVECOUNT, COUNTER;
  FLOAT FLAG, TOTAL;
  COUNTER=0;
  FLAG=0;
  FOR (MOVECOUNT=1-MOVING.LEN; MOVECOUNT<STATIONARY.LEN; MOVECOUNT++)
    {TOTAL=0;
      FOR (STAYCOUNT=0; STAYCOUNT<MOVING.LEN; STAYCOUNT++)
        IF ((MOVECOUNT+STAYCOUNT>=0) && (MOVECOUNT+STAYCOUNT<STATIONARY.LEN))
          TOTAL+=STATIONARY.VAL[MOVECOUNT+STAYCOUNT]*MOVING.VAL[STAYCOUNT];
      STORE->VAL[COUNTER]=TOTAL;
      IF (FABS(TOTAL)>FLAG)
        FLAG=FABS(TOTAL);
      COUNTER+=1;
    }
  STORE->LEN=COUNTER;
  STORE->BIG=FLAG;
}

/*****
/*
/*  USES THE MERUE WAVE-SHAPING FILTER PROCESS TO COMPUTE THE NEXT FILTER
/*  TO BE USED BY THE PROGRAM, ALSO ADDS TO WAVEBASE AND SUBFILTS
/*
/*
*****/

VOID MAKEFILTER()

{SERIES FILTER;
  ALTERNATE(WAVEBASE, &FILTER);          /* MAKE NEW FILTER(N)      */
  CONVOLVE(WAVEBASE, FILTER, &TEMP);      /* GETS NEW WAVEBASE      */
  PUTIN(&WAVEBASE, TEMP);                 /* MAKES NEW WAVEBASE     */
  CONVOLVE(SUBFILTS, FILTER, &TEMP);      /* GETS NEW SUBFILTS     */
  PUTIN(&SUBFILTS, TEMP);                 /* MAKES NEW SUBFILTS    */
  CONVOLVE(SUBFILTS, FILTER, &TEMP);      /* MAKES NEW FILTER      */
  GRAPH(FILTER, 0);
  CONVOLVE(WAVELET, FILTER, &TEMP);      /* MAKES RESULTANT SERIES */
  GRAPH(TEMP, 1);
}

/*****
/*
/*  ACCEPTS A SERIES AND RETURNS ANOTHER SERIES VARIABLE WITH THE SIGNS OF
/*  ALTERNATE NON-ZERO TERMS CHANGED
/*
/*
*****/

VOID ALTERNATE(SERIES OLDBASE, SERIES *NEWBASE)

{INT COUNTER, FLAG;
  FLAG=0;
  COUNTER=0;
  WHILE (COUNTER<OLDBASE.LEN)
    {IF (OLDBASE.VAL[COUNTER]!=0)
      {IF (FLAG)
        {NEWBASE->VAL[COUNTER]=OLDBASE.VAL[COUNTER];
          FLAG=0;
        }
      ELSE
        {NEWBASE->VAL[COUNTER]=OLDBASE.VAL[COUNTER]*(-1);
          FLAG=1;
        }
      }
    }
  ELSE

```

```

        NEWBASE->VAL[COUNTER]=OLDBASE.VAL[COUNTER];
        COUNTER+=1;
    }
    NEWBASE->LEN=OLDBASE.LEN;
}

/*****
/*
/* PUTS THE VALUES OF ONE SERIES VARIABLE INTO ANOTHER VARIABLE
/* FOR USE AS A TEMPORARY VARIABLE
/*
/*
*****/

VOID PUTIN(SERIES *NEWSER, SERIES OLDSER)

{INT COUNTER;
  FOR (COUNTER=0; COUNTER<OLDSER.LEN; COUNTER++)
    NEWSER->VAL[COUNTER]=OLDSER.VAL[COUNTER];
  NEWSER->LEN=OLDSER.LEN;
}

/*****
/*
/* LOOP CONTROL FOR MAKING FILTERS, CHECKS IF USER WANTS TO CONTINUE WITH
/* FILTER MAKING PROCESS FOR THE CURRENT INPUT
/*
/*
*****/

VOID NUMFILTS()

{CHAR CH;
  CH=' ';
  WINDOW(1,1,32,7);
  PRINTF(" PRESS SPACE BAR FOR NEXT FILTER,\N");
  PRINTF(" S TO STOP.\N");
  WHILE (CH==' ')
    {MAKEFILTER();
     CH=GETCH();
     WHILE ((CH!=' ') && (CH!='S'))
       CH=GETCH();
     IF ((TEMP.LEN+WAVEBASE.LEN-1>600) && (CH==' '))
       {TC();
        CH='S';
       }
    }
  TC();
}

/*****
/*
/* GRAPHS THE LATEST FILTER OR THE LATEST CONVOLUTION OF THE
/* FILTER AND WAVELET, DEPENDING ON INTEGER ARGUMENT
/*
/*
*****/

VOID GRAPH(SERIES WAVE, INT GRATYPE)

{INT LOCAT, COUNTER, WIDTH;
  FLOAT SCALE;
  SETVIEWPORT(10,129+GRATYPE*105,630,229+GRATYPE*105,1);
  CLEARVIEWPORT();
  SETFILLSTYLE(1,63);
  BAR(1,1,620,100);
  SETFILLSTYLE(1,7);

```

```

    BAR(3,3,618,98);
    SETFILLSTYLE(1,1);
    BAR(10,50,610,50);
    BAR(10,5,10,95);
    LOCAT=WAVE.LEN/2;
    IF (WAVE.LEN>300)
        WIDTH=1;
    ELSE WIDTH=2;
    SCALE=(48.0)/WAVE.BIG;
    WINDOW(1,1,80,25);
    IF (GRTYPE==0)
        GOTOXY(13,9);
    ELSE GOTOXY(13,25);
    PRINTF("SCALE FACTOR=%1.6F", SCALE);
    SETCOLOR(58);
    FOR (COUNTER=0; COUNTER<WAVE.LEN; COUNTER++)
        LINE(WIDTH*(COUNTER-LOCAT)+300,50,WIDTH*(COUNTER-LOCAT)+300,50-WAVE.VAL[COUNTER]*SCALE);
    WINDOW(1,1,32,7);
}

/*****
/*  GRAPHS THE INPUT SERIES (GIVEN WAVELET AND DESIRED WAVELET)
*****/

VOID STARTGRAPH(SERIES WAVE, INT GRTYPE)

{INT FLAG, COUNTER, WIDTH;
  FLOAT SCALE;
  SCALE=(48.0)/WAVE.BIG;
  IF (GRTYPE==0)
      {SETVIEWPORT(280,15,630,115,1);
        CLEARVIEWPORT();
        SETFILLSTYLE(1,63);
        BAR(1,1,350,100);
        SETFILLSTYLE(1,7);
        BAR(3,3,348,98);
        SETFILLSTYLE(1,1);
        BAR(10,50,340,50);
        BAR(10,5,10,95);
        FLAG=0;
      }
  ELSE
      {BAR(180,5,180,95);
        FLAG=170;
      }
  SETCOLOR(58);
  FOR (COUNTER=0; COUNTER<WAVE.LEN; COUNTER++)
      FOR (WIDTH=0; WIDTH<3; WIDTH++)
          LINE(4*COUNTER+10+FLAG+WIDTH,50,4*COUNTER+10+FLAG+WIDTH,50-WAVE.VAL[COUNTER]*SCALE);
}

/*****
/*
/*  INITIALIZES THE PROGRAM FOR A NEW SET OF DATA FOR GIVEN AND DESIRED,
/*  MAKES WAVEBASE AND FILTRASE
/*
*****/

VOID STARTUP()

{INT COUNTER;
  SETBKCOLOR(1);
  WINDOW(1,1,80,25);
  TC();
}

```

```

GOTOXY(1,1);
PRINTF("LENGTH OF THE WAVELET (1-40)?\N");
PRINTF("DEFAULT=40.\N");
SCANF("%D", &WAVELET.LEN);
TC();
IF (WAVELET.LEN>40)
    WAVELET.LEN=40;
PRINTF("ENTER THE %D VALUE(S).\N", WAVELET.LEN);
FOR (COUNTER=0; COUNTER<WAVELET.LEN; COUNTER++)
    {SCANF("%F", &WAVELET.VAL[COUNTER]);
     IF (FABS(WAVELET.VAL[COUNTER])>WAVELET.BIG)
        WAVELET.BIG=WAVELET.VAL[COUNTER];
    }
TC();
PRINTF("LENGTH OF DESIRED WAVE (1-40)?\N");
PRINTF("DEFAULT=40.\N");
SCANF("%D", &DESIRED.LEN);
TC();
IF (DESIRED.LEN>40)
    DESIRED.LEN=40;
PRINTF("ENTER THE %D VALUE(S).\N", DESIRED.LEN);
FOR (COUNTER=0; COUNTER<DESIRED.LEN; COUNTER++)
    {SCANF("%F", &DESIRED.VAL[COUNTER]);
     IF (FABS(DESIRED.VAL[COUNTER])>DESIRED.BIG)
        DESIRED.BIG=DESIRED.VAL[COUNTER];
    }
TC();
SUBFILTS.LEN=1;
SUBFILTS.VAL[0]=1;

/*MAKE WAVEBASE*/
FOR (COUNTER=0; COUNTER<WAVELET.LEN; COUNTER++)
    TEMP.VAL[COUNTER]=WAVELET.VAL[WAVELET.LEN-COUNTER-1];
TEMP.LEN=WAVELET.LEN;
CONVOLVE(WAVELET, TEMP, &WAVEBASE);

/*MAKE FILTBASE*/
CONVOLVE(TEMP, DESIRED, &FILTBASE);
WINDOW(1,1,80,25);
GOTOXY(39,1);
PRINTF("INPUT WAVELET          DESIRED WAVELET");
STARTGRAPH(WAVELET, 0);
STARTGRAPH(DESIRED, 1);
GOTOXY(4,9);
PRINTF("FILTER");
GOTOXY(4,25);
PRINTF("OUTPUT");
}

/*****
/*  CLEARS THE CURRENT TEXT WINDOW                               */
*****/

VOID TC()
{
    TEXTCOLOR(1);
    CLRSCR();
    TEXTCOLOR(7);
}

/*****
/*  ASKS USER IF HE WANTS TO REPEAT THE PROCESS AND INPUT NEW SETS OF DATA  */
*****/

```

```

/*****/

INT CONT()

{CHAR CH;
  PRINTF(" TO GRAPH ANOTHER PRESS SPACE BAR,\N");
  PRINTF(" PRESS Q TO QUIT.\N");
  WHILE ((CH!=' ') && (CH!='Q'))
    CH=GETCH();
  IF (CH==' ')
    RETURN(1);
  ELSE
    RETURN(0);
}

/*****/
/* MAIN BLOCK-SETS UP GRAPHICS, LOOPS THROUGH FUNCTIONS */
/*****/

MAIN()

{INT CON, GDRIVER=DETECT, GMODE=EGAH;
  INITGRAPH(&GDRIVER, &GMODE, "");
  CON=1;
  WHILE (CON)
    {STARTUP();
     NUMFILTS();
     CON=CONT();
    }
  WINDOW(1,1,80,25);
  TEXTCOLOR(0);
  CLRSCR();
  TEXTCOLOR(3);
  SETBKCOLOR(0);
}

```



## Phase Recognition

The purpose of this experiment was to see if a neural net could recognize the phase shift of a waveform regardless of the waveform's frequency. The test began by creating a data file of time samples from waves of twenty different frequencies and in sixteen different phase shifts for a total of 320 facts for the net to learn from. Next the facts were randomized.

Randomizing facts proved to decrease the amount of time required for a net to learn those facts presented to it. By doing this, the net adjusts the weights for the data as a whole, not in groups. For example, if the data were left in order, the different frequencies would be grouped together with the sixteen phase shifts in order as well. As the net would try to train from this it would train on one frequency at a time. After it passed one frequency the weights would be adjusted for that frequency only. When it approached a new frequency the weights would be changed around again to satisfy only that new frequency. Although the net would eventually train off of the ordered facts, training time for randomized facts was found to be dramatically shorter.

After the facts were randomized, the data was properly organized for use by the Brainmaker software and training was begun. Soon after, the net had been successfully trained. Now the true test would take place by creating a new data file of different frequencies in various phases. The network successfully recognized the phase of each new wave proving that a neural net can recognize the phase shift of a waveform regardless of it's frequency.

## Frequency Combination/Recognition

The purpose of this experiment was to see if a neural net could recognize the component sine waves of a complex wave, and eventually recognize them with the addition of noise. For the first try only four frequencies were used; 1, 2, 4, and 8 Hz. Each series of values was made by taking 64 samples of a wave created combining any of the four frequencies. The data was input to the Brainmaker software and the net was successfully trained. Training time was relatively short.

The next step would be to use six frequencies; 1, 4, 7, 10, 13, and 16 Hz. Again, the net trained, taking more time due to the increase in facts. Now, eight frequencies would be used, 1, 2, 3, 4, 5, 6, 7, and 8 Hz. The net trained successfully yet the time required was much greater than that for the other tests. Naturally this was due to the even larger amount of facts; the number of facts increased exponentially. As time became noticeable, training efficiency became a factor. It was observed that adjusting the learn rate of the net could decrease the amount of training time.

The learn rate is basically the magnitude of the changes made to the weights. When the net makes an error, the weights are adjusted in hopes of getting a better response the next time that fact is encountered. Eventually the weights are adjusted to get the correct response every time, yet if the weights are changed too drastically, the correct value may never be found.

At the start of training the learn rate would be set high. This allowed the net to quickly adjust to the general values

required. The net would soon begin to get some facts correct.

After a while though, the increase in correct responses would level off, while the net would not yet be fully trained. It was found that by lowering the learn rate at this point, the net could "tune in" better. By making less dramatic changes to the weights, the net could find the correct positions faster.

Now that the nets were fully trained on the clean waves, noise was added to the data. The results of the tests were as follows:

Frequencies	Correct/Total	Percentage
4	16/16	100%
6	62/64	97%
8	241/256	94%

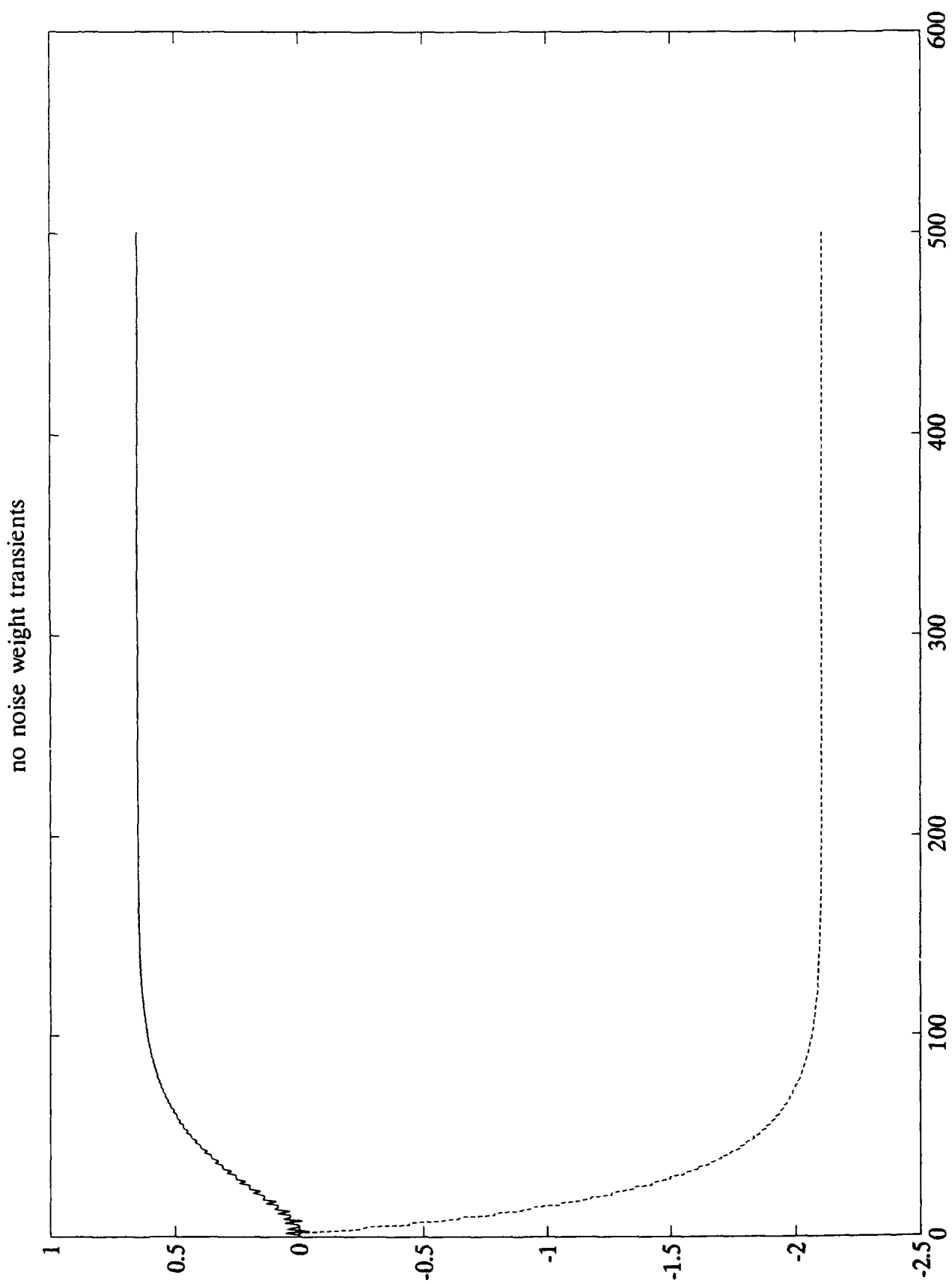
A possible source of error would be that the net required more samples of the complex waves, but the number of samples was limited by the computers memory. The more frequencies used, the more difficult it would be for the net to distinguish the components. Therefor, a more detailed series of samples might have produced better results.

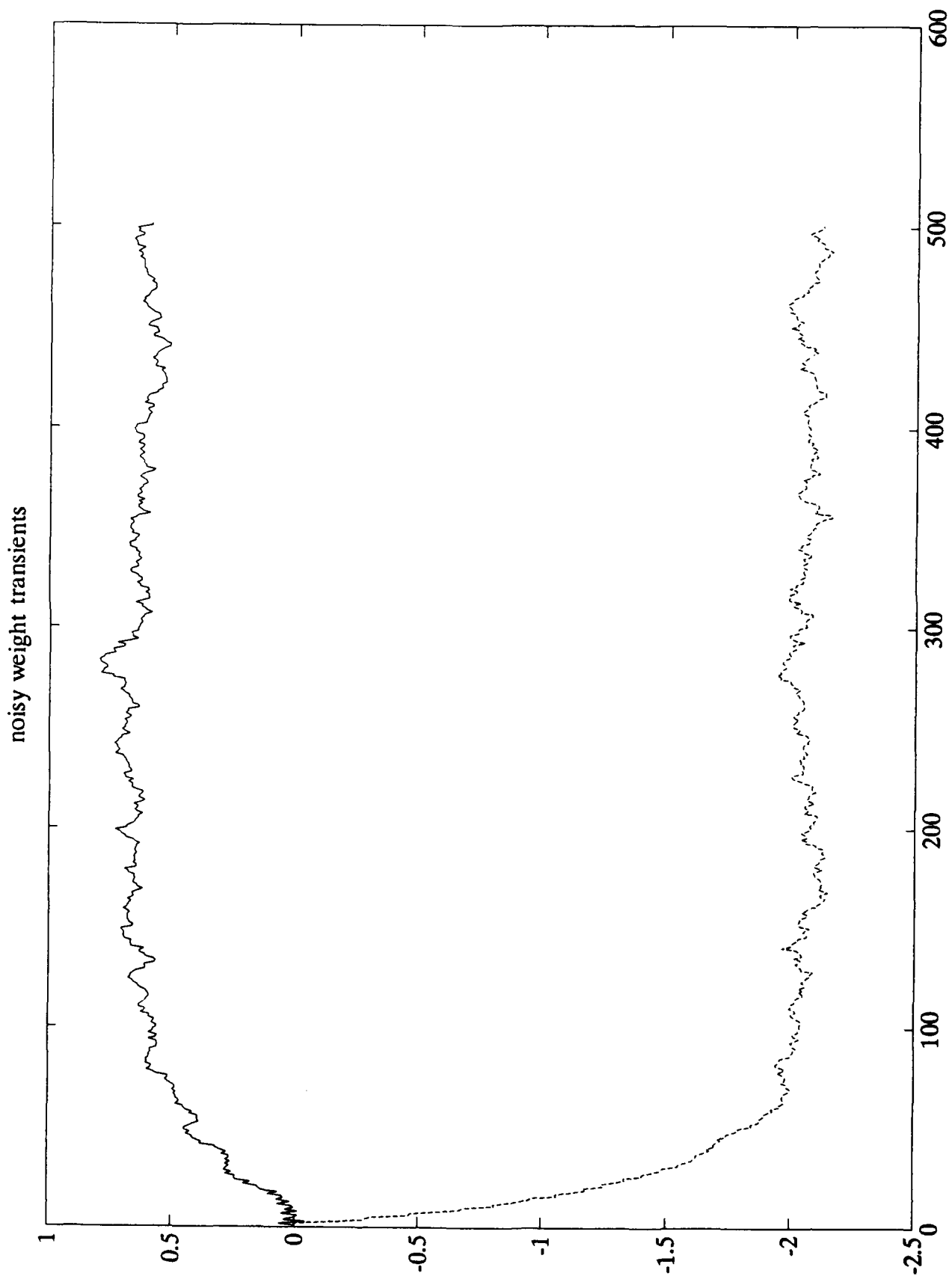
## Perceptron Program

Another program assigned to me was to make a multi-layer perceptron which would train through a back-propagation algorithm while using a sigmoid neuron function. By doing this, understanding neural nets as a whole became much simpler.

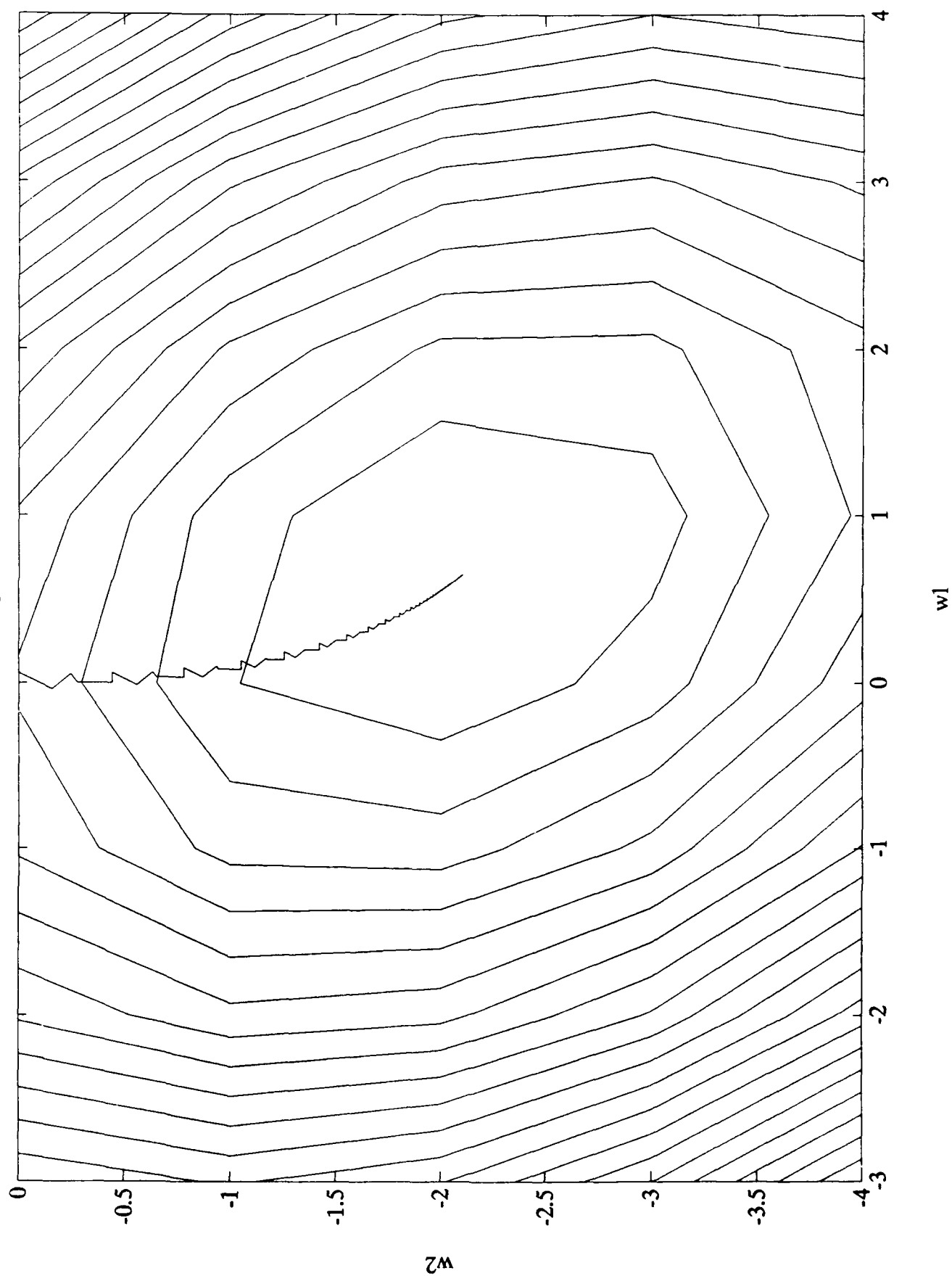
## Adaptive Filters

Some of my time at Rome Labs was spent learning about adaptive filtering. With the help of Lt. Goldstein, I studied the effects of noise on weight transients. Although the noisy filter would need endless re-adjustment, it could be seen that the relative time for converging was the same for both noisy and non-noisy signals. The following graphs illustrate the effects of the added noise.

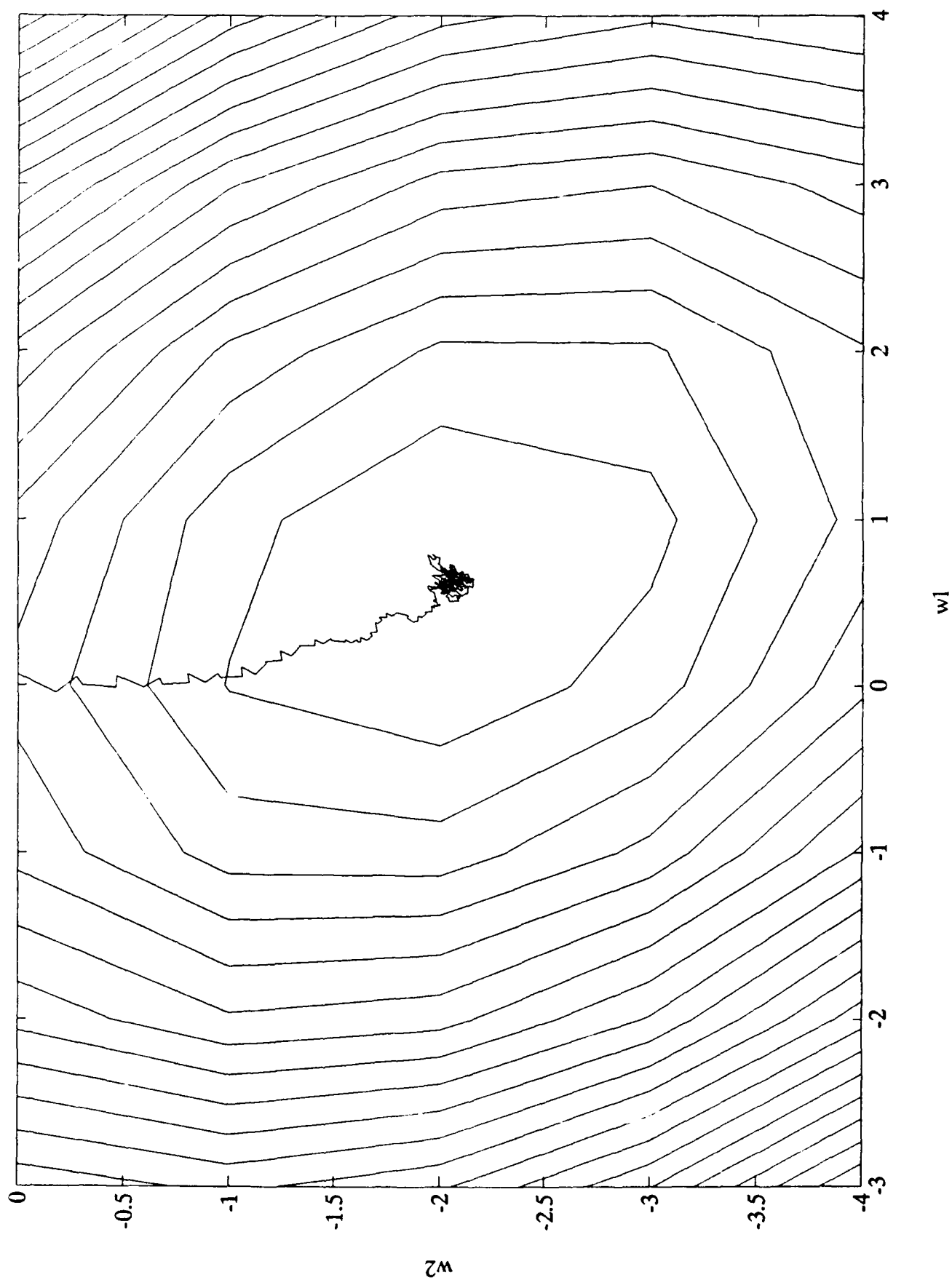




no noise walk down performance function



noisy walk down performance function





### Works Cited

Brainmaker. California Scientific Software. Sierra Madre, CA:  
1990.

Mereu, R. F., 1976. "High Resolution time-series analysis with an  
error-free wave-shaping filter of finite length." Phys. Earth  
Planet. Inter., 12: 232-236.



# Prototyping on SuperCard



Joe Senus  
RL/IRDO  
(315) 330-7791  
16 Aug 91



• Try to learn and use the MAXI applications on the Macintosh Workstation in order to be able to create rapid prototyping visualization environments.

• Be able to visualize and prototype. Be able to capture the methodology and prototype a part of the MAXI application on the latest workstation environments.



?



- Student Background

- Objective

- Environment

- Methodology

- Results



High School

- Presently attending high school
- Previous experience with science
- Have not had any previous work experience, but hope to return next year
- Hope to major in a science oriented subject in the future, particularly physics

Macintosh

210,800

Voice Mail

118 in. Color Imaging

Color Imaging

18 in. Hi-Fi Acceptor Stereo VCR

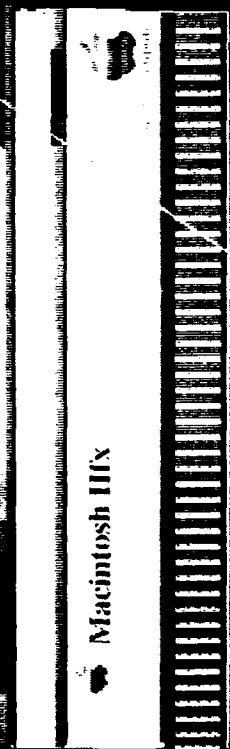
Two Panasonic VCR's

Speakers

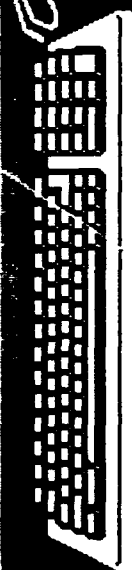


19 in. Color monitor

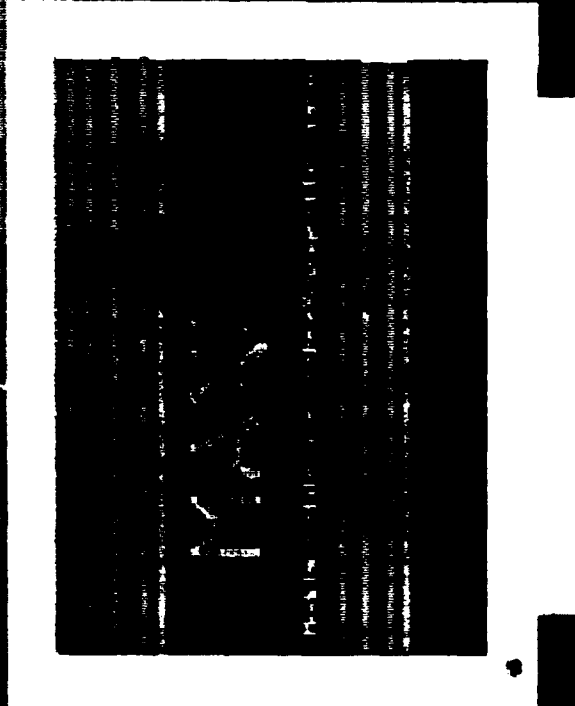
Navigator



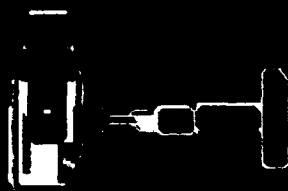
Macintosh IIIx



210 and 80 megabyte  
hard drives

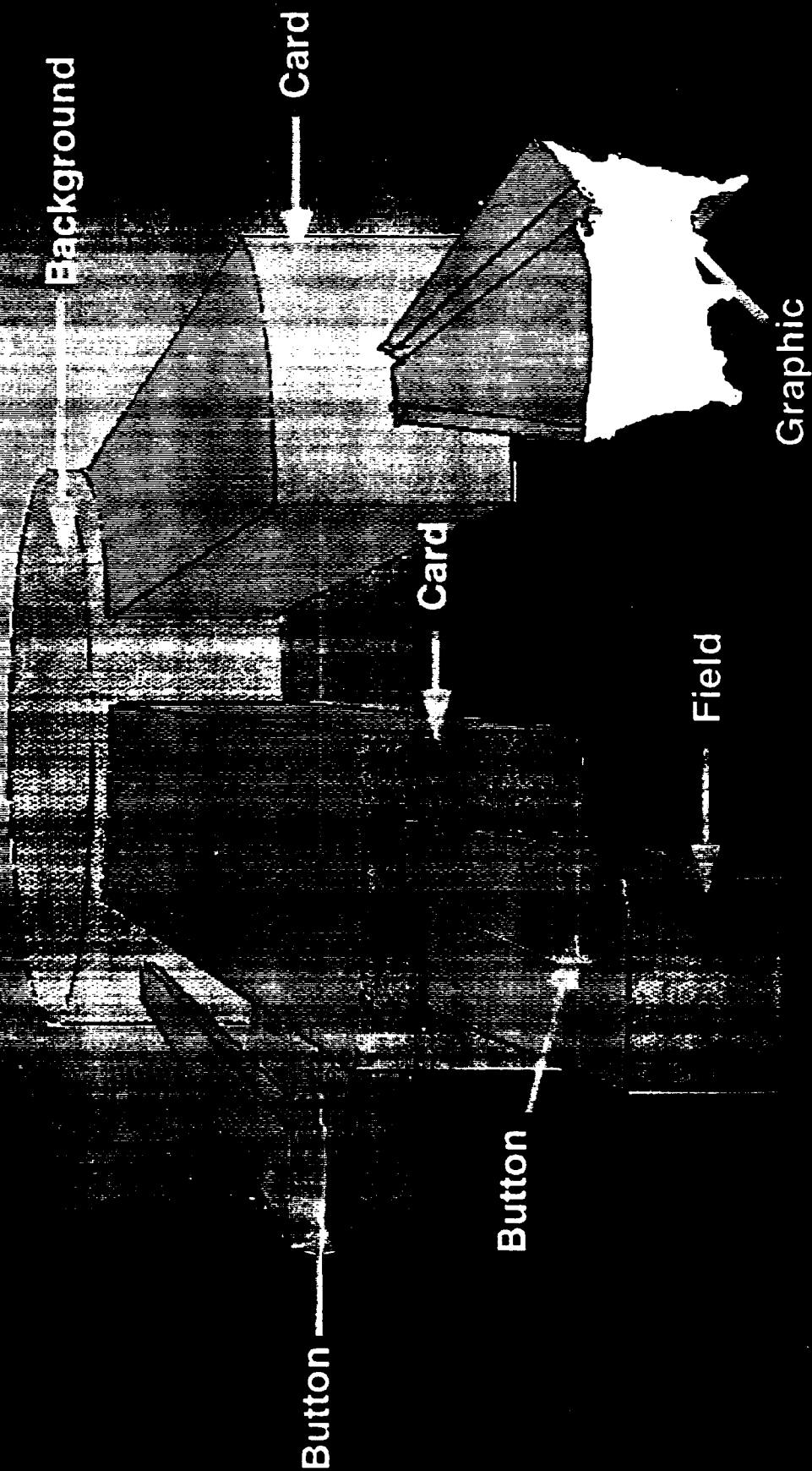


Panasonic camera



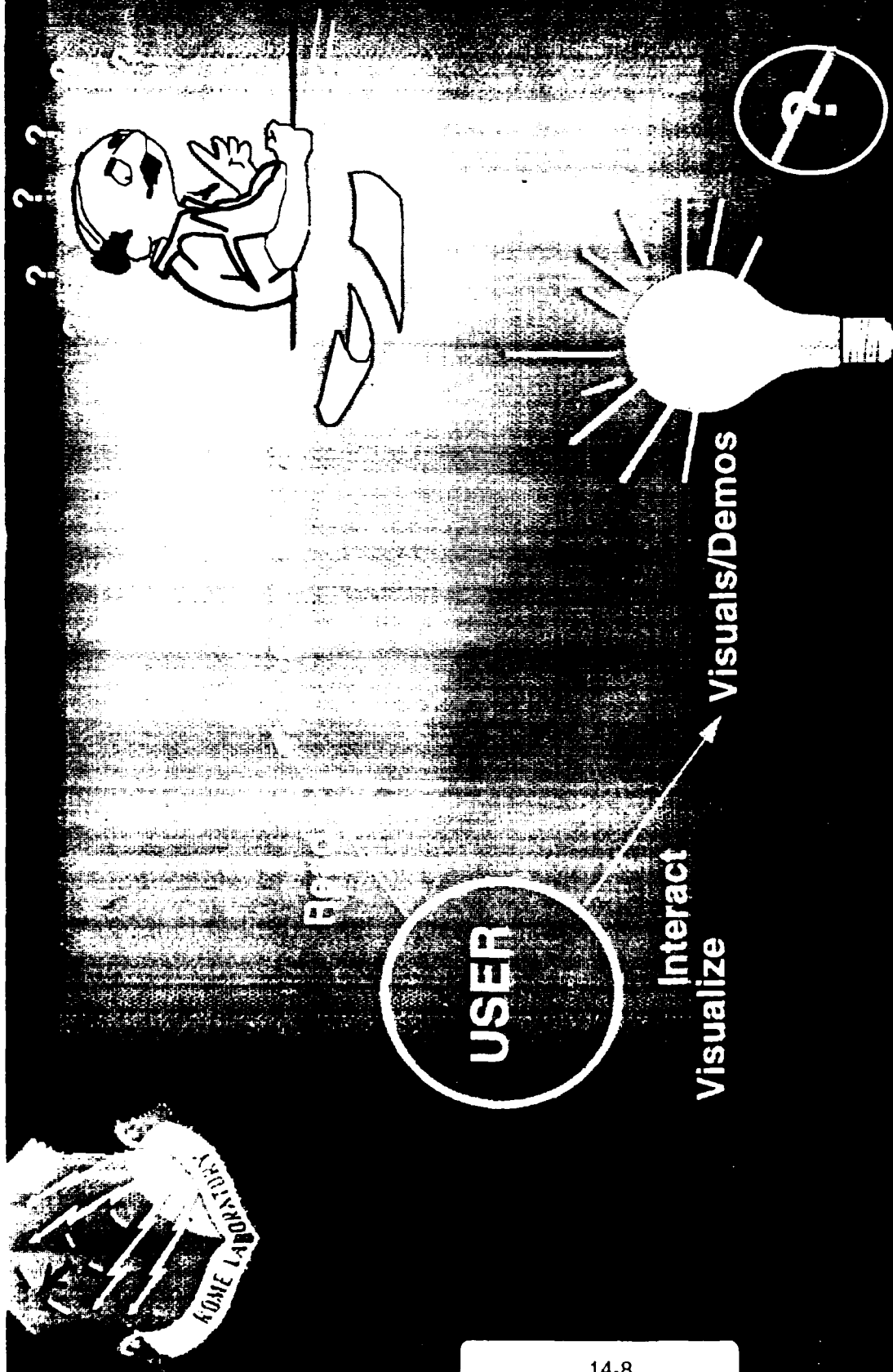


# SuperCard - The steps in creating a SuperCard Project



Prototyping Software Tool Environment





## Advantages of Prototyping Visualization

Idea by Daniel Peck

## OBJECTIVE:

- Requirements
  - The User tells the Builder what he wants, for example, "I want a system that will..."
- Concept of System
  - The system Builder now tells the User what he can do with the system. The User wants...
- Flow Chart
  - The system Builder now creates a flow chart of the procedures/ or actions...
- SuperCard Prototype
  - The system Builder then creates a working demo on SuperCard which can give the User a visualization of what he's looking for...
- Show Prototype
  - The Builder can now show off his prototype and receive comments and ideas for possible changes
- Agreement
  - The Builder and User finally agree on the implementation for what you want to build

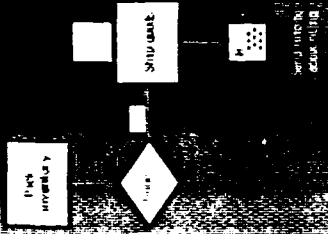


Technology of Concept of Operation Prototyping...

Requirements

It's an automated intelligence system that can do that which I think I want

Flow Chart



"This is what they want it to do"...

"I want an Automated Intelligence System"

"This is how I want it to flow"...

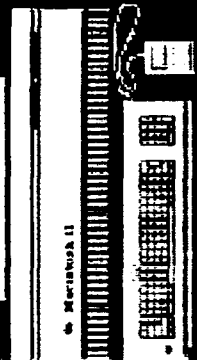
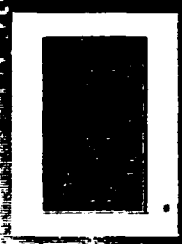
Show Prototype

THIS STINKS!!!

Agreement



Basic Prototype



"If all goes good, I'll pay you now..."

"I want to see it work and I want hands on demonstrations..."



# This is an example of a Flow Chart

This is the MAXI Menu hierarchy

## MAXI Menu

System Menu

New

Memo

Status

Message Queue

Manning

Op5COM

Sign on

Sign off

Circuit

Activate/  
Allorate

Deactivate/  
Deallocate

Quit

Activity Menu

Message Queue

INDICOM Queue

Work File

Message File Query

Hold Queue

Build message

Message Profile

Addressee List

Report Format

Op5COM

Exercise Support

Selection of the Work File application gives the user access to the stored workfiles. Selection of this option gives the user a choice of Access(ing), Retrieve(ing), and/or Rename(ing) the available/accessible workfiles.

When the user arrives at the MAXI Menu he will have a choice between the System Menu and the Activity Menu. In this particular case we will be going to the Activity Menu and then choosing the Work File application.

1) Access option indicates to the user which subareas workfiles are accessible from the current subarea.

2) Retrieve option produces a dialog box soliciting the user's selection of a specific workfile for retrieval.

3) Rename option enables the user to change the subjects under which the indicated workfile is indexed.



An example of a working SC prototype...

# MAXI Message Review

Unclassified

TRUSTED  
PATH

System

Activity

Help

## Activity Status

Message Queue	4
INDICOM Queue	5
Work File	8
Message File Query	4
Hold Queue	total: 20

## Notices:

Not signed on... unable to change classification

# This is an example of a MAXI Queue Status Review done on SuperCard...

Unclassified

File Edit

25-JULY-91 13:11:25		QUEUE STATUS REPORT				
SUEAREA	CRI	FLC	IN4	PRI	ROU	TOTAL #MESS
CMOIN	0	0	0	0	0	0
CMOIND	0	0	0	4	0	4
CMOINL	0	0	0	2	0	2
CMOINO	0	0	0	2	0	2
CMOINW	0	0	0	0		
CMOINX	0	0	0	1		
CMOSSA	0	0	0	1		
CMOVTA	0	0	0	2		
CMRMS	0	0	0	1		
CMCHAT	0	0	0	2		
CMCSUM	0	0	0	1		
CMDFLT	0	0	0	0		
CMDIRF	0	0	0	2		
CMEEUR	0	0	0	1		
CMEXRM	0	0	0	1		
CMEXER	0	0	0	0		
CMILSTX	0	0	0	2		
CMOINO	0	0	0	2		
CMOINA	0	0	0	1		
CMEXTO	0	0	0	0		
CMBHUT	0	0	0	0		
						22-JUN-90 16:56
						22-JUN-90 16:44
						22-JUN-90 16:44



## Advantage

### Building a Prototype

- Better understanding
- Better communication

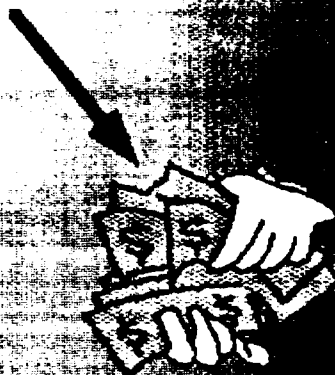
"I think we should try this"



+



=



Wasting time and money

When you don't communicate you waste money



+



=



## RESULTS

- Became familiar with COTS Software applications
- Explored the Sun/Sparc Workstations development environment, including programming, and UNIX functions.
- Learned about how to use SuperCard, and started to prototype a small section of MAXI on it.
- Captured prototyping methodology.



# **1991 HIGH SCHOOL APPRENTICESHIP PROGRAM**

**KEN BARNES**

**MENTOR: CHARLES SMITH, CALSPAN**

**AEDC**

**AUGUST 2, 1991**

*TITLE: MODIFICATION OF COST ESTIMATION PROGRAMS*

### **ABSTRACT**

The budget for each wind tunnel project conducted in the Propulsion Wind Tunnel (PWT) facility is estimated by computer programs. The 4 foot transonic (4T) wind tunnel needed an estimation program like the 16 foot transonic and supersonic (16T/S) wind tunnels. The assignment over this eight weeks of the apprenticeship program was to create a cost estimate program for the 4T tunnel. The project was divided into two phases. First, completed 4T tunnel tests were researched and cost rates were derived from them. Then, a 4T program was derived from an existing 16T/S program. The result was a new estimate program for the 4T tunnel.

## **INTRODUCTION**

During operations of the Propulsion Wind Tunnel (PWT) facility, large amounts of time and money are consumed. To plan the PWT budget, a computer program is used to estimate the cost of each test project. However, there was no estimate program available for the 4 foot transonic (4T) wind tunnel. The purpose of this paper is to explain how the 4T estimate program was developed based on the already existing 16 foot transonic/supersonic (16 T/S) program.

## **PROCEDURES**

### **Research---**

Even though the 16T/S and the 4T programs are structured similarly, most cost calculations are different. Converting the inputs and calculations from the 16T/S program to the 4T program presented difficulties. In addition, the 16T/S program was in need of revision.

Program research includes determining what each equation in the program was for, and whether it needed updating. Then, appropriate 4T experts were consulted for the information needed to update constants and equations. These equations included: manhour wages plus surcharge, computer processing hours per user occupancy hours (CPU/UOH), megawatt hours per air-on hours (MHW/AOH), and additional material cost.

Also, the Air Force contracting office recently sent a memo updating the costs per fiscal year for manhours, surcharges, natural gas, electricity, and computer hours. Documentation of AEDC wind tunnel test statistics and computer changes were obtained.

### **Development--**

Extra options were added to the program that were not available in the 16 T/S program which applies to fiscal year (FY) 91 estimates only. An option was inserted to select any of the next 5 fiscal years for the 16T/S and 4T program. Variables in the program depend on the FY input. (See *Fiscal Year Inputs*.) This feature allows for a more accurate estimate of future FY projects. The 16T/S and 4T program must be revised every year. The new algorithms of the 4T program were written in such a manner that program verification was simplified. Two costs were added to the printout: materials, and maintenance surcharge, neither of which were in the original program. (See *The Printout*.)

The cost in the original estimate program was partially determined by test types such as inlet tests, nozzle/afterbody tests, or balance tests. The new 4Testimate program uses test complexity for the rates. The more complex test requires higher manhour costs. A separate option was added to the choices of test complexity for the Captive Trajectory Store (CTS) test.

Subtle errors in the complex code required considerable time to correct. Numerous checkpoints were calculated, and program performance was compared to actual wind tunnel tests costs which were published in a statistical survey.

### **RESULTS**

The new program, *4T Cost Estimation*, gives the 4T tunnel an estimation program available to only the 16T/S tunnel before. The process of comparing the program output to actual test results assured program accuracy. The new program was made available and used immediately by PWT project engineers.

### OBSERVATIONS

The RDL Apprenticeship Program presents many opportunities for experience and learning. Apprentices have the chance to participate as well as observe actual tests. They are a help to engineers instead of a hindrance. Computers are a vital part of any workplace. This program puts AEDC's best computers at the Apprentice's disposal. Computer literacy is almost mandatory for these jobs.

In the eight weeks spent at AEDC, much time was spent gaining experience on wind tunnel projects to support development of the 4T estimate program. These experiences included support in the 16T wind tunnel, computer programming for software other than the cost estimate program, and an informal study of fluid dynamics. The RDL summer program provides the greatest opportunities for high school students to gain experience in the workfield of engineers.

# THE PRINTOUT

## TUNNEL 4T PRELIMINARY COST ESTIMATE

DATE:  
ESTIMATOR: KEN BARNES  
UOH: 200  
AOH: 150  
TEST TYPE/COMPLEXITY: SIMPLE  
MACH RANGE: 0.25-0.39  
RE X(10-6): 2.5  
HIGH PRESSURE AIR REQUIRED  
SECURITY GUARD REQUIRED  
ANALYSIS REQUIRED

NOTE: The numbers on this sheet are not actual estimates

FISCAL YEAR: 91

PHASE	MANHOURS	MANHOUR \$
1	100	201
2	150	420
3	50	100
4	1000	2012
5	1976	4012
6	4500	9689
7	1400	3000
8	1070	2162
 CALSPAN	 10146	 19297
 SSI	 50	 100
 TOTAL	 10196	 19397
 MAINTENANCE SURCHARGE		 44673

### RESOURCES:

ITEM	UNITS	\$
ELECTRICITY	4000	41000
COOLING WATER	200	1024
NATURAL GAS	200	1112
VAX	30	1001
AMDAHL	31	1300
MATERIAL		2000

TOTAL ESTIMATED PROJECT COST: \$ AROUND 800,000

## FISCAL YEAR INPUTS

```
40 PRINT "4T COST ESTIMATION PROGRAM"
41 REM INPUTS
50 INPUT "INPUT FY (91-96)"; FY
60 IF FY = 91 THEN CAL = XX.XX: SUR = X.XX: GR = X.XX: ER = XX.XX: SSI = XX.XX:
VAX = XXX: AMR = XXX
61 IF FY = 92 THEN CAL = XX.XX: SUR = X.XX: GR = X.XX: ER = XX.XX: SSI = XX.XX:
VAX = XXX: AMR = XXX
62 IF FY = 93 THEN CAL = XX.XX: SUR = X.XX: GR = X.XX: ER = XX.XX: SSI = XX.XX:
VAX = XXX: AMR = XXX
63 IF FY = 94 THEN CAL = XX.XX: SUR = X.XX: GR = X.XX: ER = XX.XX: SSI = XX.XX:
VAX = XXX: AMR = XXX
64 IF FY = 95 THEN CAL = XX.XX: SUR = X.XX: GR = X.XX: ER = XX.XX: SSI = XX.XX:
VAX = XXX: AMR = XXX
65 IF FY = 96 THEN CAL = XX.XX: SUR = X.XX: GR = X.XX: ER = XX.XX: SSI = XX.XX:
VAX = XXX: AMR = XXX
70 IF FY > 96 THEN GOTO 75
71 IF FY < 91 THEN GOTO 75
72 GOTO 80
75 GOSUB 2400: GOTO 50
79 PRINT
80 INPUT "INPUT THE PROJECT TITLE--->...etc."
```

FY = FISCAL YEAR  
CAL = CALSPAN MANHOUR DOLLAR RATES (EFFORT B)  
SUR = MAINTENANCE SURCHARGE  
GR = NATURAL GAS DOLLAR RATES  
ER = ELECTRICITY DOLLAR RATES  
SSI = SCHNEIDER MANHOUR DOLLAR RATES  
VAX = VAX DOLLAR RATES (PER CPU HR)  
AMD = AMDAHL DOLLAR RATES (PER CPU HR)

## ACKNOWLEDGEMENTS

Thanks to RDL, Major D.C. Hart (AEDC), and 1Lt. Greg Nordstrum (AEDC).

Thanks to the engineers who helped with this assignment and showed what it was like being an engineer-- Mentor Charlie Smith, Reggie Riddle, Darhl Frazier, Harry Kaupp, Bob Useton, Pete Lauer, Keith Silvus, Lori Holt, and John Dempsey.



## ***BIBLIOGRAPHY***

### **GW-Basic Made Easy**

**Bob Albrecht and Don Inman**

**printed by: Osborne McGraw-Hill**

**Berkeley, California, 94710**

**copyright 1989**

1991 AIR FORCE OFFICE OF SCIENTIFIC RESEARCH  
(AFOSR)  
HIGH SCHOOL APPRENTICESHIP PROGRAM

**Direct Write Scene Generator  
Laser-Beam Mode Analysis**

Kevin Belew

Mentor: Sid Steely, Calspan

9 August 1991

**ABSTRACT**

This research was conducted in relation to the Direct Write Scene Generator method of testing. The objective of the research was to illustrate that the laser beam size can be determined from the classical limits of the corresponding isotropic two-dimensional quantum mechanical harmonic oscillator. In particular the laser beam spot size can be determined from the corresponding harmonic oscillator's classical limits. In addition it is shown that the fraction of the laser beam energy contained within the classical limits approaches unity in agreement with the correspondence principle.

## INTRODUCTION

The evaluation and testing of complex optical sensor systems is important in the development of a space-based defense system. These optical sensor systems are used for detection, discrimination, tracking, and kill assessment functions. As the problems that limit sensor performance are solved, testing the complex sensor systems effectively becomes more difficult. In order to evaluate the sensor systems, we must be able to create realistic mission scenes that can be input to the sensor systems. Work at AEDC involves two different techniques for creating scenes. The first and more traditional method exploits conventional blackbody technology. The second and more advanced method uses lasers and a Direct Write Scene Generator (DWSG) to simulate complex scenes. The DWSG deflects a laser beam to irradiate individual pixels of a Focal Plane Array (FPA) in order to create a scene for mission simulation as illustrated in Fig.1.

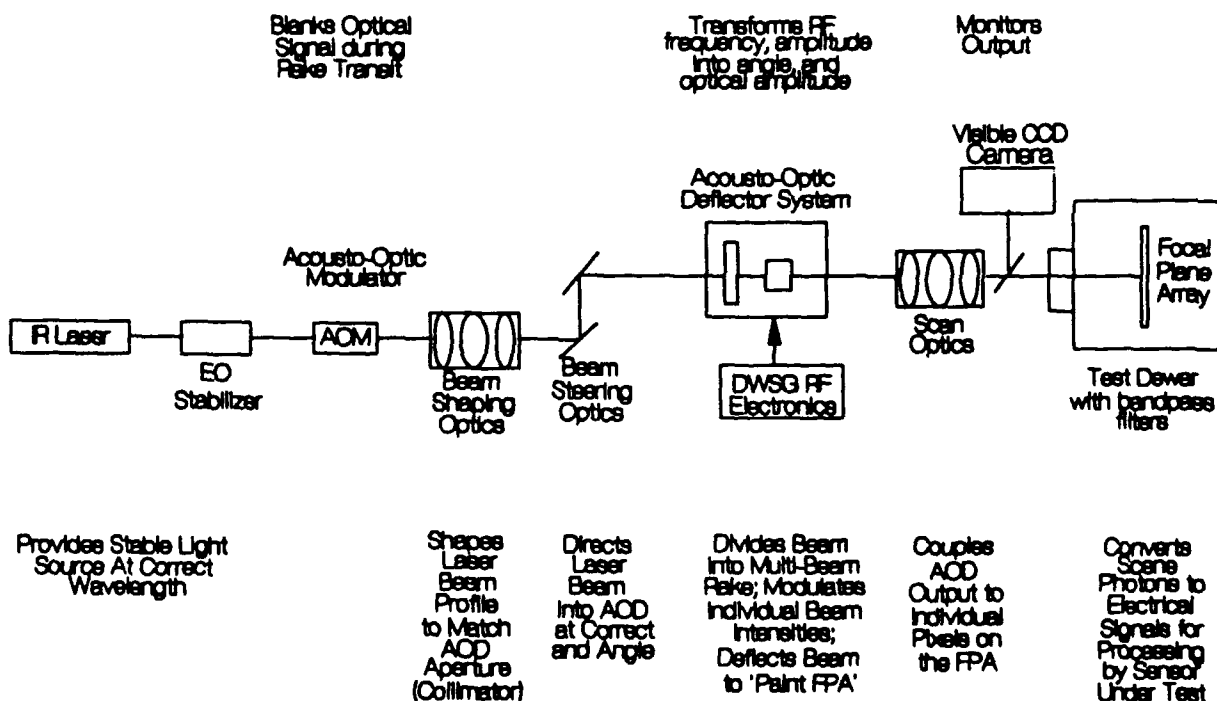


Figure 1. DWSG Optical Schematic

A commercially available two-axis acousto-optic (AO) Bragg cell deflects the laser beam. The drive electronics which control the AO devices are capable of creating, editing, and displaying the contents of the scenes sent to the deflection device. The output is then directed through an optical window and bandpass filter to the FPA, which is housed in a cryogenic test dewar. The DWSG is, therefore, analogous to a television in that it uses the integration time of the detector and creates complex images with a steered electron energy beam as illustrated in Fig. 2. Yet, the DWSG differs from a television in that the energy source is a beam of photons, which is deflected and intensity-modulated by acousto-optic devices. The AO cells are controlled by a complex radio frequency (RF) electronic system that converts the scene to the proper control signals. The photon beam then irradiates the focal plane directly instead of a projection screen [Lowry, 1991].

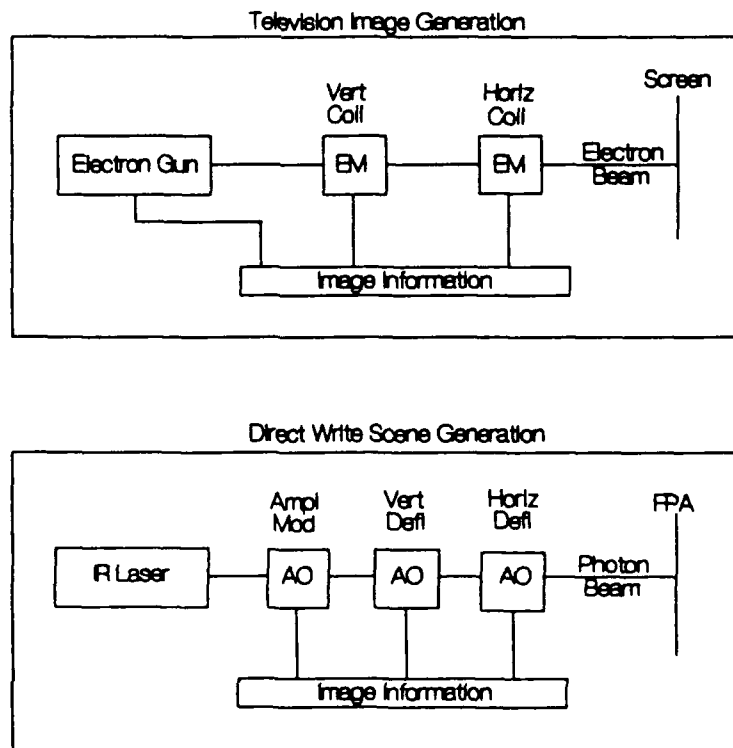


Figure 2. Television-DWSG analogy.

The ability to provide complex scenes on a pixel-by-pixel basis is a major strength of the DWSG system. Gamma events and other highly dynamic situations can be simulated by single-pixel frame-to-frame changes. The scene content is not constrained by the physical hardware of the scene generator because of the DWSG programmability. Because the output is computer controlled and is capable of simulating motion of the sensor platform, the technique works correspondingly well with staring or scanning sensors. The use of lasers as the radiative source enables narrow spectral bandpass filters to be used to suppress ambient radiation background that would normally enter the dewar assembly and raise the noise floor. Also, many FPA evaluations can be performed with the test article in a small dewar rather than a more costly and larger test chamber [Lowry, 1991].

Since the DWSG method uses a laser, the stability and coherence of the laser is important in minimizing the energy fluctuations and noise on the FPA. The mode of the laser is also of importance because it is a factor in determining the spot size, fractional energy within the spot size, and the FPA pixel being irradiated. The lasers used in the DWSG are designed to be very stable and to operate in single mode. The laser beam conditioning/expansion optics are designed to expand the laser beam in order to optimally fill the AO cell, providing a small Gaussian irradiated spot on the FPA pixels. Any laser beam mode instabilities will alter this optimum beam expander and AO cell mode of operation, leading to inherent and undesirable fluctuations in the fraction of energy collected by the AO cell and subsequently focused onto the FPA pixel by the scan optics.

My particular area of research is concerned with the relationship between the fractional energy contained in the spot size of a Hermite-Gaussian laser beam versus its beam mode and, primarily, how results from the quantum mechanical harmonic oscillator problem can be applied in understanding this basic relationship. In particular, it was illustrated that the laser beam size can be determined, in analogy, from the classical limits of the corresponding isotropic two-dimensional quantum mechanical harmonic oscillator. Also during my eight-week work period, I was introduced to a new scientific tool, Mathematica, which I also helped to Beta test. The Mathematica program provided an excellent tool to support scientific research and scientific visualization. Other work was also conducted using the Mathematica program but is not included in this paper.

## DISCUSSION AND RESULTS

Since my objective was to research the relationship between the spot size of a Hermite-Gaussian laser beam and the classical limits of a quantum mechanical harmonic oscillator, I first familiarized myself with my objectives, tasks, and relevant literature. Next I tried to become as proficient as possible at using the computer and the various programs which I would be using (i.e. MS-DOS, Mathematica, Mathtype, Winword, and Fastcad). Having reviewed the software needed, I was then ready to begin my research.

The first problem was to grasp the correlation and relationship between the classical harmonic oscillator and the quantum mechanical harmonic oscillator which has the same functional form as a TEM Laser

intensity profile. The probability density graphs illustrated in Figure 3 were generated using Mathematica. The harmonic oscillator probability densities help to visually see and better understand the relationship. The dotted curves in Fig. 3 and Eq. (1) represent the classical harmonic oscillator probability density

$$p_c(x) = \frac{1}{\pi(x_0^2 - x^2)^{1/2}} \quad (1)$$

where  $x_0 = (2m + 1)^{1/2}$ .

The solid curves in Fig. 3 and Eq. (2) represent the one-dimensional quantum mechanical harmonic oscillator probability density

$$p_q(x) = H_m^2(x) \exp(-x^2). \quad (2)$$

The vertical lines in Fig. 3 indicate the classical limits  $x_0$  of the harmonic oscillator problem. It is easy to see from Fig. 3 that the average of the peaks and valleys of the quantum mechanical probability densities asymptotically approaches the classical density for large  $m$  in agreement with the correspondence principle. All of the peaks are also seen to be contained within the classical limits of the harmonic oscillator as is being proved in general by Sturm's theorem. For large  $m$  the quantum mechanical harmonic oscillator is less likely to be observed outside of the classical limits and is more likely to be observed within the classical limits, as anticipated by the correspondence principle.

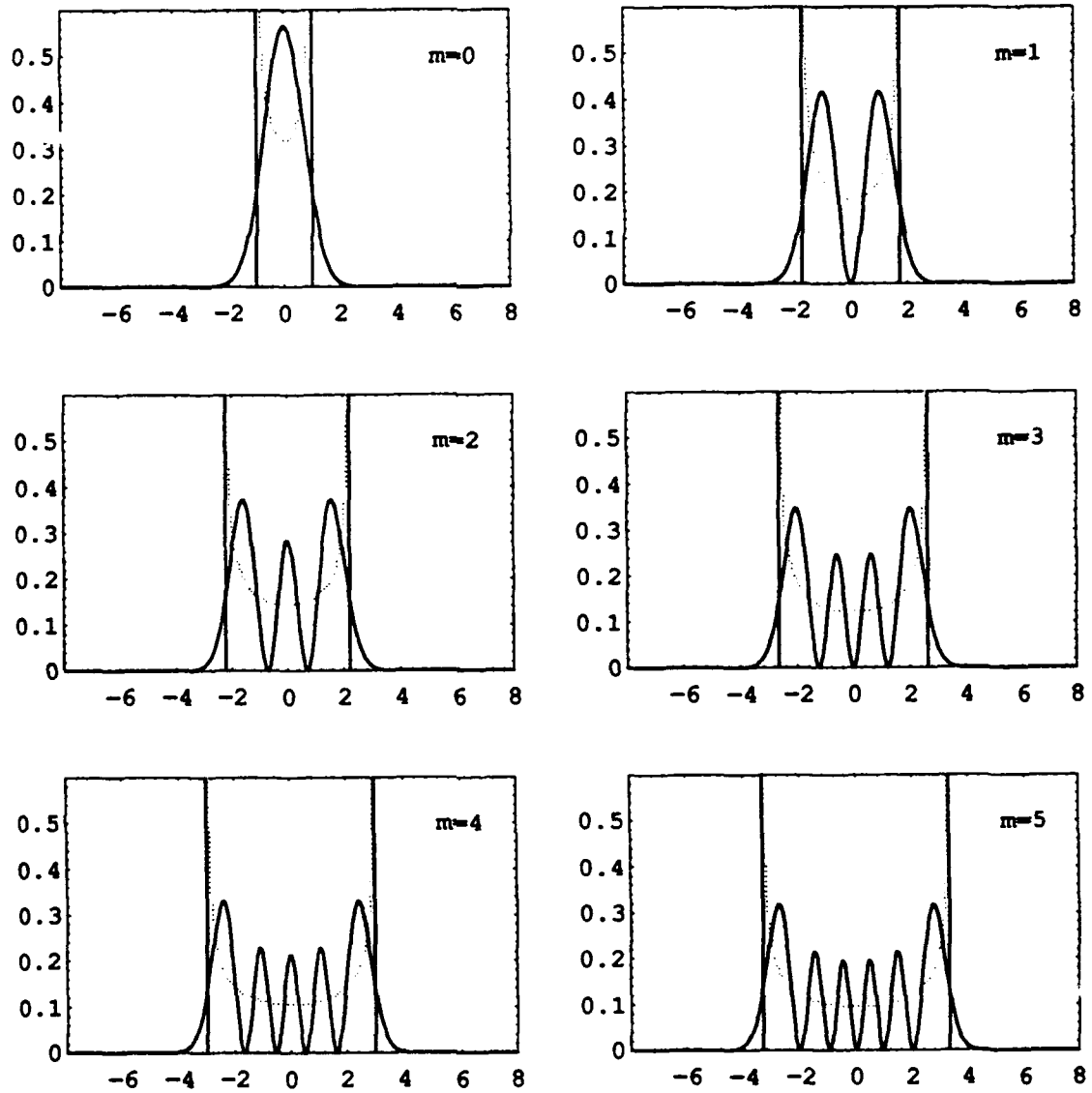


Figure 3. Harmonic oscillator probability densities.



The next step was to understand the correlation between the harmonic oscillator problem and the classically limited fractional energies of TEM Hermite-Gaussian laser beam mode intensities. The two-dimensional quantum mechanical harmonic oscillator and the Hermite-Gaussian laser beam's relative intensity profile are of the same functional form as illustrated by the solid curves in Fig. 3 and by Eq. 3 (Carter, 1980).

$$E(x, y, z) = E_0 \frac{\sigma_x(0)\sigma_y(0)}{\sigma_x(z)\sigma_y(z)} H_m^2\left(\frac{x}{\sigma_x(z)}\right) H_n^2\left(\frac{y}{\sigma_y(z)}\right) \exp\left(-\frac{x^2}{\sigma_x^2} - \frac{y^2}{\sigma_y^2}\right) \quad (3)$$

where the "1/e beam waist" along the z-axis of the beam is

$$\sigma_s(z) = \sigma_s(0) \left(1 + \frac{z^2}{z_{0,s}^2}\right)^{1/2}, \quad (4)$$

$$z_{0,s} = \frac{2\pi}{\lambda} \sigma_s^2(0), \quad (5)$$

and twice the variance of the irradiance (for s equal to x and y) is

$$\sigma_s^2(z)_t = \frac{2 \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} s^2 E(x, y, z) dx dy}{\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} E(x, y, z) dx dy} \quad (6)$$

Similar to the harmonic oscillator problem, standard normalization is used to define the fractional energy as defined in Eq. (7).

$$f_{m,n} = \frac{\int_{-\sqrt{2m+1}}^{\sqrt{2m+1}} H_m^2(\xi) \exp(-\xi^2) d\xi \int_{-\sqrt{2n+1}}^{\sqrt{2n+1}} H_n^2(\eta) \exp(-\eta^2) d\eta}{2^{m+n} \pi m! n!} \quad (7)$$

The relative energy contained within the "spot" of the laser is defined as the fractional energy within the classical limits because a large fraction of the energy is contained within these limits. Thus it can be deduced that the fractional energy of a Hermite-Gaussian laser beam, as defined by the classical limits, asymptotically approaches unity as the beam mode is increased, in agreement with the correspondence principle.

Equation (7) and Mathematica were used to write a program to integrate the TEM laser beam intensities for  $TEM_{m,n}$  modes with  $m,n$  ranging from 0 to 200 with results tabulated in Table 1. The Mathematica results indicate that the fractional energies within the spot sizes increase with the beam modes, which is contrary to some published references [Carter,1980 and Oshea,1985]. Only the diagonal  $m,m$  modes were then evaluated from 200-1000. The data from this integration were consistent with the first set of data in that the fractional energy asymptotically approached unity in both cases.

Table 1. Fractional Energies for TEM Laser Beam Modes

	0	1	2	3	4	5	6	7	8	9	10
0	0.71	0.792	0.802	0.807	0.81	0.812	0.814	0.816	0.817	0.818	0.819
1	0.792	0.883	0.894	0.9	0.903	0.906	0.908	0.91	0.911	0.912	0.913
2	0.802	0.894	0.906	0.911	0.915	0.917	0.919	0.921	0.922	0.923	0.924
3	0.807	0.9	0.911	0.917	0.92	0.923	0.925	0.927	0.928	0.929	0.93
4	0.81	0.903	0.915	0.92	0.924	0.927	0.929	0.93	0.932	0.933	0.934
5	0.812	0.906	0.917	0.923	0.927	0.929	0.931	0.933	0.934	0.935	0.936
6	0.814	0.908	0.919	0.925	0.929	0.931	0.933	0.935	0.936	0.937	0.938
7	0.816	0.91	0.921	0.927	0.93	0.933	0.935	0.937	0.938	0.939	0.94
8	0.817	0.911	0.922	0.928	0.932	0.934	0.936	0.938	0.939	0.94	0.941
9	0.818	0.912	0.923	0.929	0.933	0.935	0.937	0.939	0.94	0.941	0.942
10	0.819	0.913	0.924	0.93	0.934	0.936	0.938	0.94	0.941	0.942	0.943

One of my tasks was to plot the asymptotic curve of the fractional energies of Hermite-Gaussian laser beam modes. Mathematica was used to write a program to calculate the fractional energies of the  $TEM_{m,n}$  modes ranging from  $TEM_{0,0}$  up to  $TEM_{1000,1000}$ . I used Mathematica to also plot the data as illustrated in Figure 4.

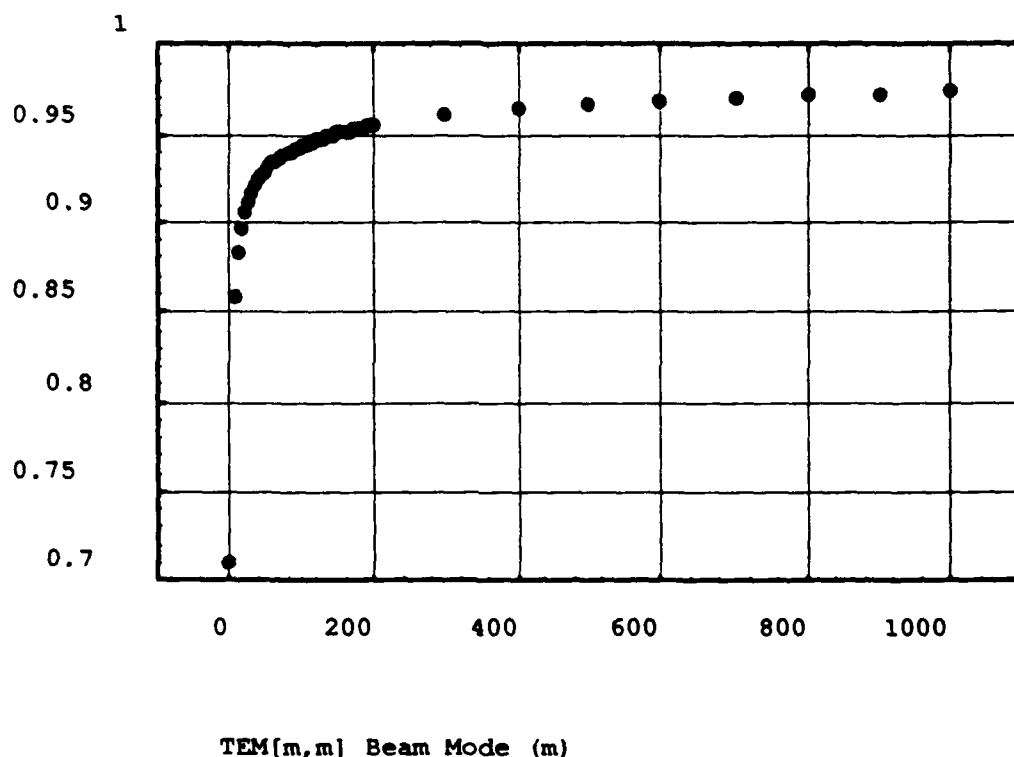


Figure 4. Fractional beam energies for Hermite-Gaussian laser beams.

## SUMMARY

Work this summer was to investigate the relationship between the fractional energy contained in the spot size of a Hermite-Gaussian laser beam versus its beam mode and how the quantum mechanical oscillator problem can be applied to better understand potential laser beam mode fluctuations. In particular, it was to be illustrated that the laser beam spot size can be determined from the classical limits of the corresponding isotropic two-dimensional quantum mechanical harmonic oscillator. During my eight-week work period, I made extensive use of the Mathematica program.

## REFERENCES

- Carter, William H. "Spot Size and Divergence for Hermite Gaussian Beams of any Order", Applied Optics , Vol.19, No.7, 1 April 1980.
- Lowry, H. S., P.D. Elrod, P.J. Johnson. "AEDC direct write scene generation test capabilities," SPIE, Vol.1454, 1991.
- O'Shea, Donald C. Elements of Modern Optical Design. John Wiley & Sons, Inc., New York. 1985.
- Wolfram, Stephen. Mathematica: A System for Doing Mathematics. Addison Wesley Publishing Co., Redwood City, California. 1988.

***RTR TRAILER CABLE SPOOLING SYSTEM***

***MARK COLEMAN  
TULLAHOMA HIGH SCHOOL  
VANDERBILT UNIVERSITY***

***BEN WILLIAMS, SVERDRUP, MENTOR***

## ABSTRACT

This project was designated to provide an efficient means of storing and deploying the twenty-five cables used by the <sup>Sverdrup Technology</sup> Real Time Radiography (RTR) team at <sup>Arnold Engineering</sup> ~~Sverdrup~~ <sup>Development Center</sup> Technology. These cables are an integral part of the RTR mobile x-ray lab facilities. These cables range from 150 feet to 300 feet in length and from 0.1575 inches to 1.2525 inches in diameter. Rolled and unrolled by hand, the RTR trailer cables take approximately one working day for two craftsman to either deploy for use or store after use. Many man-hours have been expended in the past to take care of the cables necessary to take data at an engine firing away from Arnold Engineering Development Center (AEDC). The spool described in this report is designed to greatly reduce the time, labor, and cost of any trip where the RTR trailer's cables would be utilized. It is estimated that this system would save twenty man-hours per test installation. The design work for the spool system was completed as well as a resources chart, man-hour list, and a project schedule.

## ACKNOWLEDGMENTS

First of all I would like to thank all of the personnel at Technology Section EL3 for their contributions to my work and for their helpful insights. I would especially like to thank the section for being so friendly and for accepting me from the beginning as a colleague and not just a temporary high school apprentice. In particular, I would like to thank: my mentor Ben Williams, Brent Cox, Jerry Farris, Gary Storey, Tim Wallace, and Danny Catalano for their help on my spooling project; Tom Daugherty, Rick Moyer, and Robert Howard for giving me graphics projects to complete; Mike Anderson, Jim Nichols, Jeff Ready, and David Fann for their aid in my becoming acquainted with the computer software packages used here at Sverdrup; Gene Smith and Gail Arnold for their aid in faxing my pay voucher every two weeks.

## GENERAL DESCRIPTION

The first step taken towards the completion of this project was to take the dimensions of the RTR trailer storage belly. This is where the spool system is to be located. After the pertinent measurements were taken, the preliminary design was discussed with knowledgeable personnel. Along with engineering associate Brent Cox, this design was refined until it met all of the necessary requirements. This refined design was altered slightly in order to make use of stock materials readily available here at the base. Then detailed drawings of all nine spools and the spool system were completed on the LOTUS FREELANCE PLUS system, a business graphics software package. A

project resources, man-hours, and schedule were compiled and then drawn on FREELANCE.

This cable spooling system is a badly needed accessory to the existing RTR mobile x-ray facilities. The Real Time Radiography group here at ~~Sverdrup~~ <sup>AEDS</sup> periodically travels to other bases and test facilities around the nation to take x-ray data during rocket motor firings. The diagnostic information gained from these tests is very useful in assessing rocket motor performance. It can be used by manufacturers to improve motor designs or in the case of motor failures to identify the source of the problem. Trips to test sites are expensive, and any time that can be saved while on location would greatly reduce expenses. At present, it takes almost a full day to unroll the cables and the same amount of time to roll them back up. A spool system like the one described in this report would decrease the time spent rolling and unrolling the RTR trailer cables by almost two thirds. With this system, it would take approximately three hours to store and deploy the cables, thus reducing project costs.

#### **DETAILED DESCRIPTION**

The first step in the design of the RTR cable spooling system was to take measurements of the RTR trailer storage belly. A digital vernier calibrator was used to measure the diameters of the RTR cables. A tape measure was used to measure the dimensions of the RTR trailer storage belly. This data established the size limits for the spool system, the major requirement of the project. A basic design was then developed.

One of the most important steps taken to complete the final design of the cable spooling system was to determine how big the spools would have to be to accommodate all twenty-five cables. Each spool had to have an extension off one of its flanges where its associated cable could be attached to a pigtail connector hanging down from the inside of the trailer belly. To allow for access to this connector, a space of at least six inches was needed to allow ones hand to reach inside to adjust the connectors. Since space was the primary limitation, twenty-five spools with six inch extensions would be impossible. Some of the cables had to be paired on the same spool because of space limitations. The cables were matched according to size and type. Since some of the cables possessed magnetic fields and some were camera cables, they were separated so that the data carried by the camera cables would not be degraded. The cables were arranged to fit on nine different spools.

The size of each spool was then calculated mathematically. The spools had to have a four inch shaft for the cables to wrap around. However, the power cable required twelve inches to spool around because this is its minimum specified bending radius. The spool



flanges could be only 17.5 inches in diameter, leaving 0.5 inches for the pin to lock the spool in place. There was 8.5 inches of space between the spool shaft and the end of its flange. To determine the length of spool needed to house the 150 and 300 feet cables, first the diameter of the cable, or the combined cables, was recorded. This value was then divided into the 8.5 inches to determine how many rows, or wraps, the cable could make around the spool. The circumference of the circle at the top of each row was then calculated in inches. These circumferences were added and divided by twelve to give the total number of feet of cable that would spool around one column on the spool. The widths of these columns were the diameter of the cable being stored on that particular spool. The number of columns of cable on the spool was then found by dividing the length of the cable, 150 or 300 feet, by the amount of feet in each column. By multiplying the number of columns by the diameter of the cable and dividing by twelve, the length of the spool in inches was obtained. A calculator was used to do all of the mathematics.

After the spool dimensions were computed, the preliminary design was altered. Specifications were changed to meet requirements and to make use of stock materials. For example, a rim was added to the extension side flange of each spool to serve as a mount for the cable connectors. In addition, the flange thicknesses on several spools were increased by 0.25 inches because of their large size.

Once the final design was complete, detailed drawings of the spool system design were completed on the FREELANCE PLUS system.

To complete the cable spooling system design, a project resources, man-hours, and schedule were developed with the aid of my mentor, Ben Williams. The type of materials needed for the system were found in the AEDC stock catalog, and the total project materials cost was approximated. The number of man-hours spent on the project's design work were recorded. The number of man-hours needed for fabrication and installation and checkout of the system were estimated. A project schedule was also compiled, providing a late August or early September completion date .

## **RESULTS/CONCLUSION**

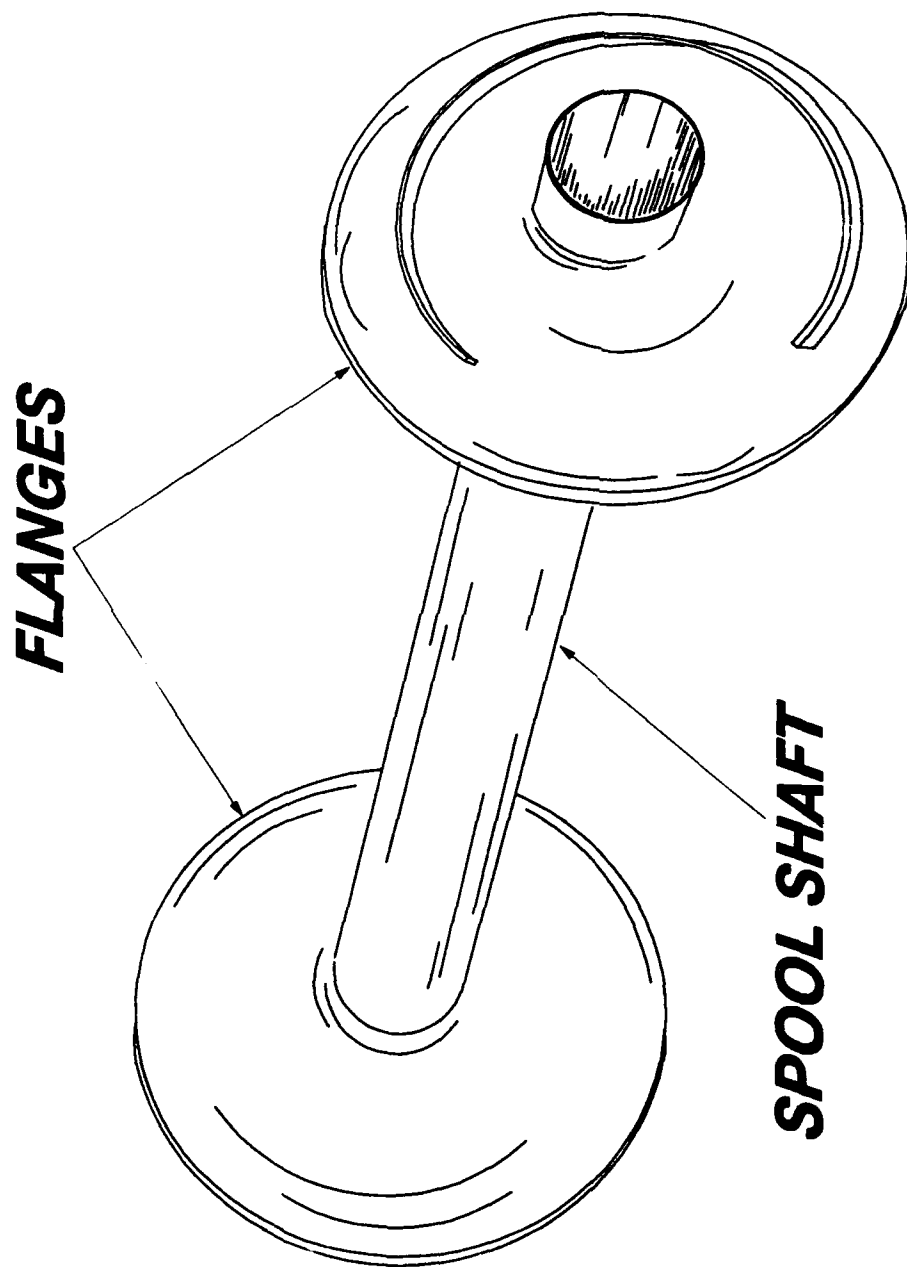
The primary result of this project was the completion of the plans that provide for a spooling system that would be a positive addition to the existing RTR mobile x-ray facilities. The compiling of the project's resources list, man-hours involved, and schedule were also accomplished. The spool design drawings and the project resources list, man-hours, and schedule are found at the end of this paper. Based on the work completed and the apparent need for a more efficient means of employing the RTR trailer cables, I

recommend that the project be continued and that the proposed RTR cable spooling system design be put into effect as soon as possible.

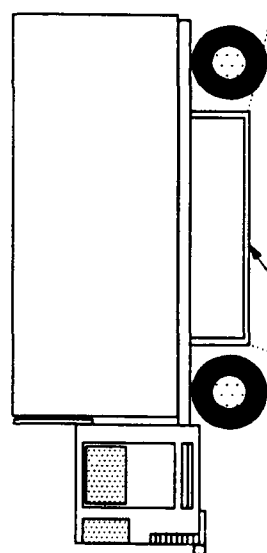
### **INSIGHTS GAINED**

I learned many things during my apprenticeship this summer. Probably the most important insight I gained was seeing first hand what a typical engineer does during the day. I realize that the atmosphere here at AEDC may differ from that elsewhere and that engineering jobs vary greatly. However, I still have a much better idea of the big picture of engineering than I did before I began working AEDC.

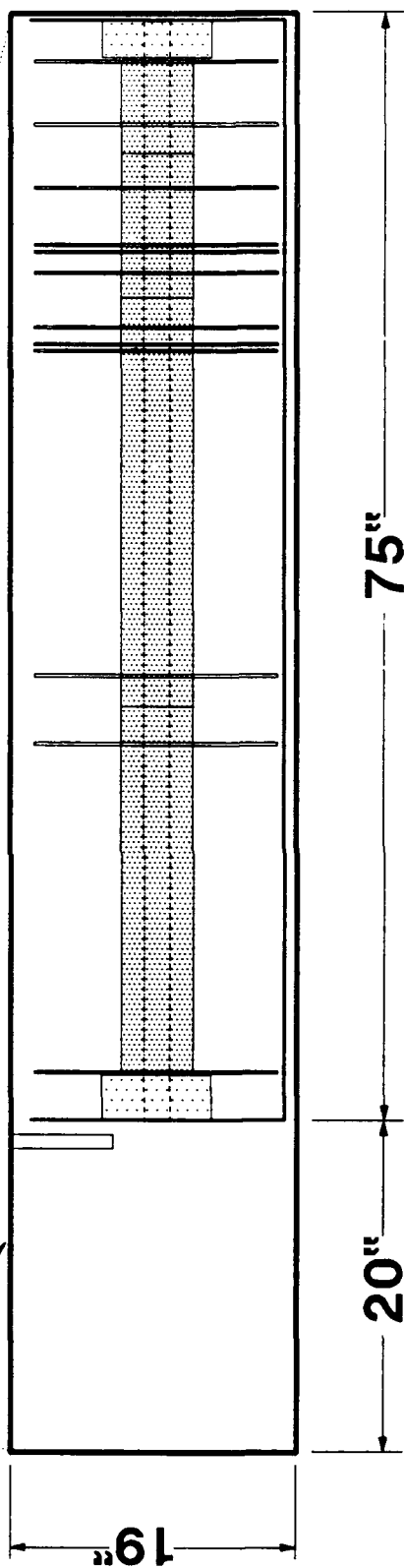
This summer I became quite familiar with the QUATTRO PRO, SPRINT, and FREELANCE PLUS software packages. SPRINT is a word processor; FREELANCE PLUS is a business graphics package; QUATTRO PRO is a spreadsheet package with graphics capabilities. I was also, for the first time, introduced to the processes of how to determine man-hours, develop a project schedule, and use a catalog to locate project materials and predict project costs. I thoroughly enjoyed these aspects of my project because they exposed me to a more complete scope of the engineering field than I would have otherwise obtained.



## **BASIC SPOOL SKETCH**



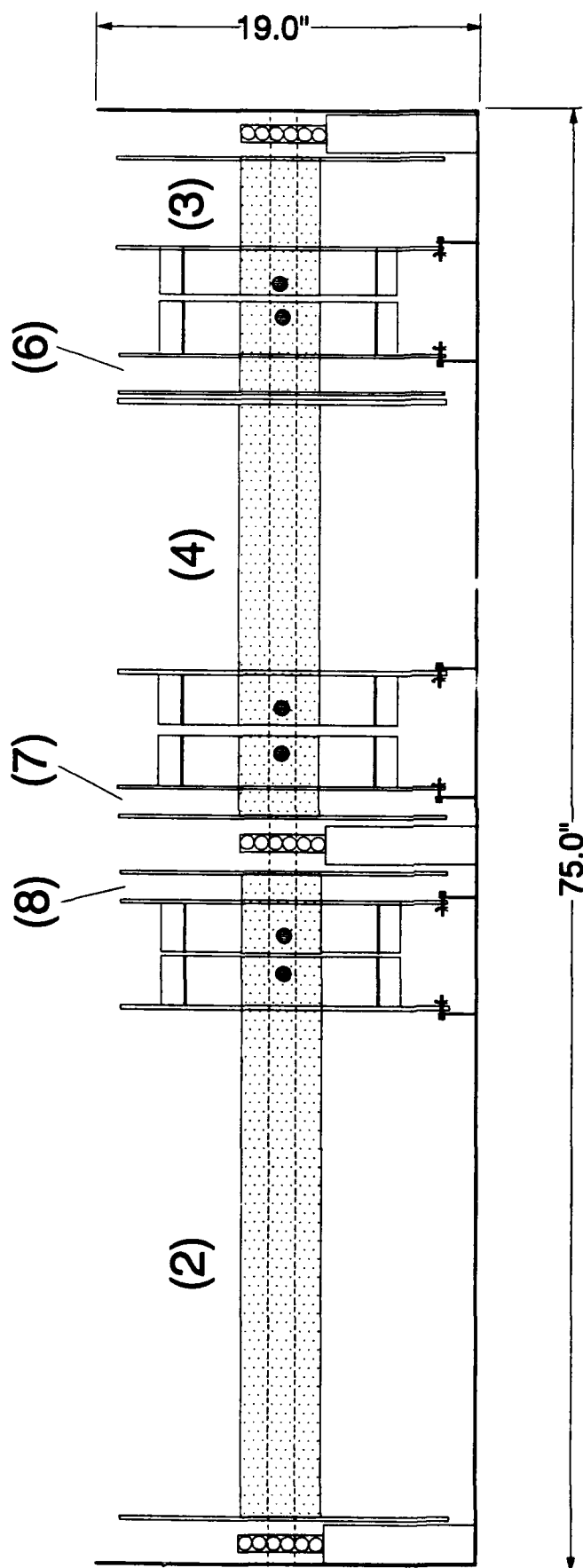
**STORAGE BELLY**



***LOCATION OF SPOOL SYSTEM IN RTR TRAILER***

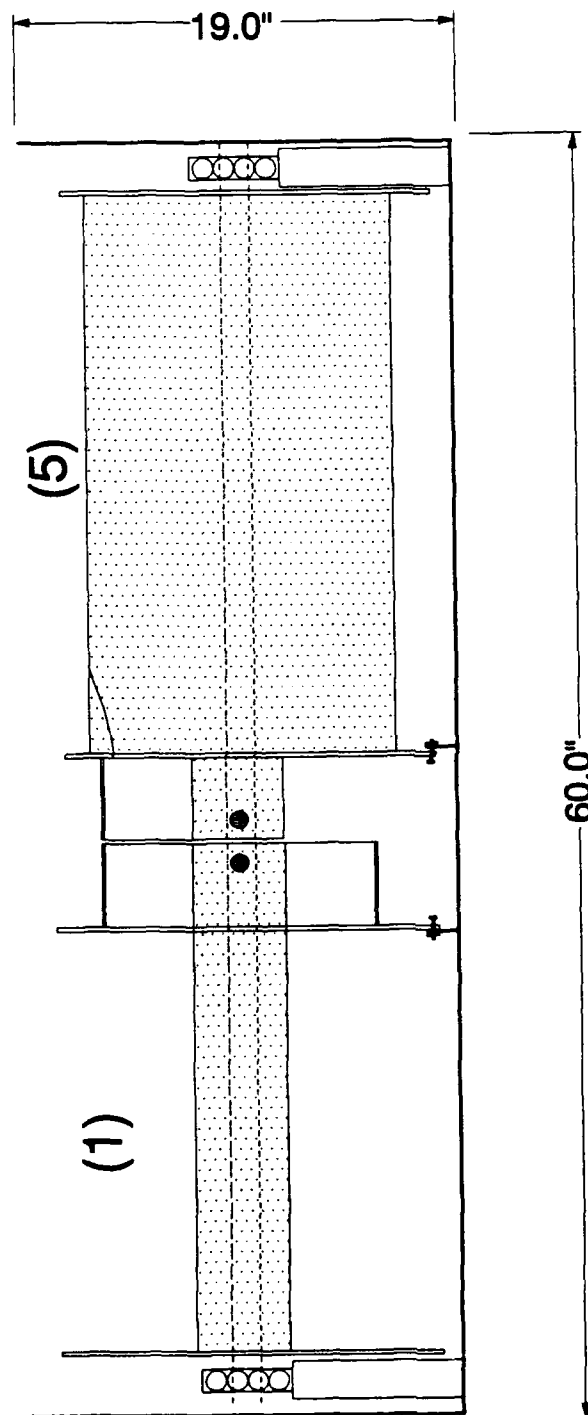
## SPOOL SPECIFICATIONS THAT DIFFER

SPOOL NUMBER	SPOOL LENGTH	WIDTH OF SPOOL SIDES	CABLES HOLDING	DIAMETER OF SPOOL SHAFT
#1	20.0"	0.25"	GREEN HOSE, BLACK HOSE	4.00"
#2	26.5"	0.25"	GREEN HOSE, RED HOSE, RG 8, MAGGIE CONTROL, (3) RG 58, INTERCOM, RS 232	4.00"
#3	4.5"	0.125"	LIGHT HORN CABLE	4.00"
#4	14.5"	0.25"	LENS CONTROL, (5) RG 59, INTERCOM	4.00"
#5	26.0"	0.25"	POWER CABLE	12.00"
#6	2.0"	0.125"	GUARD BOX, INTERCOM	4.00"
#7	1.5"	0.125"	PHONE	4.00"
#8	1.5"	0.125"	INTERCOM	4.00"
#9	10.5"	0.25"	ICACOM	4.00"



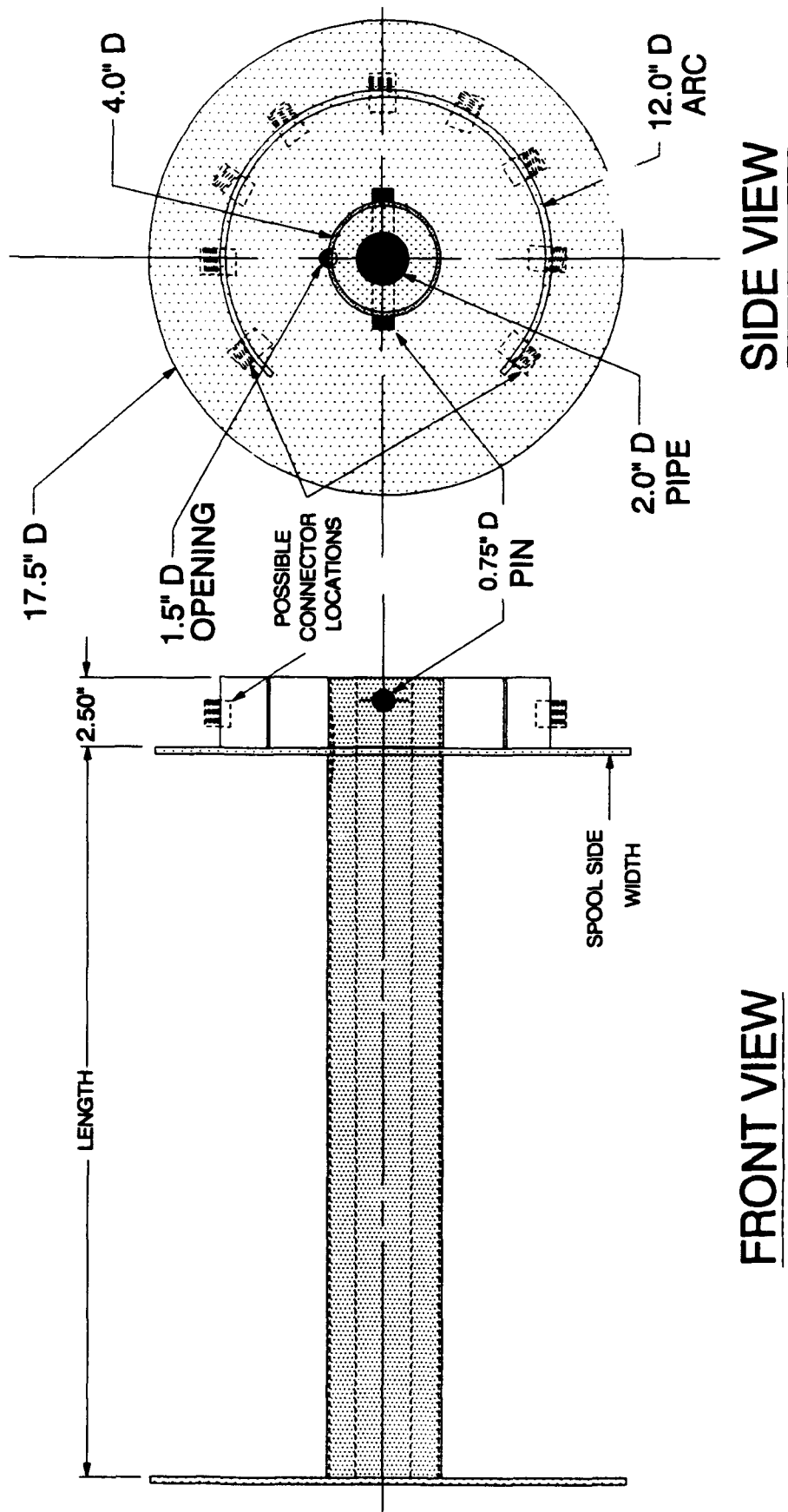
# SAMPLE SPOOL ARRANGEMENT

FOR SIDE A



## SAMPLE SPOOL ARRANGEMENT

FOR SIDE B



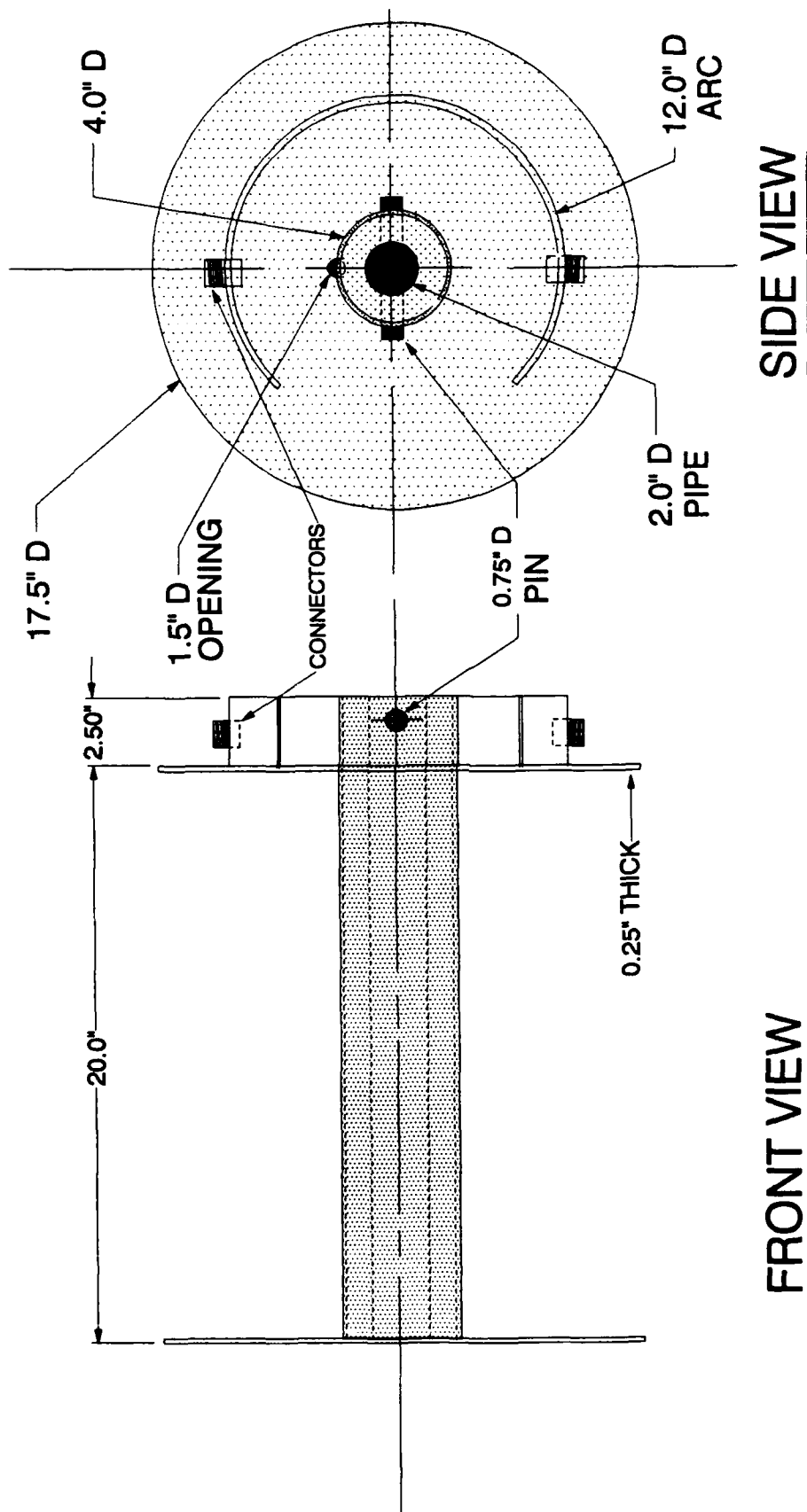
**FRONT VIEW**

**SIDE VIEW**

## **BASIC SPOOL DESIGN**

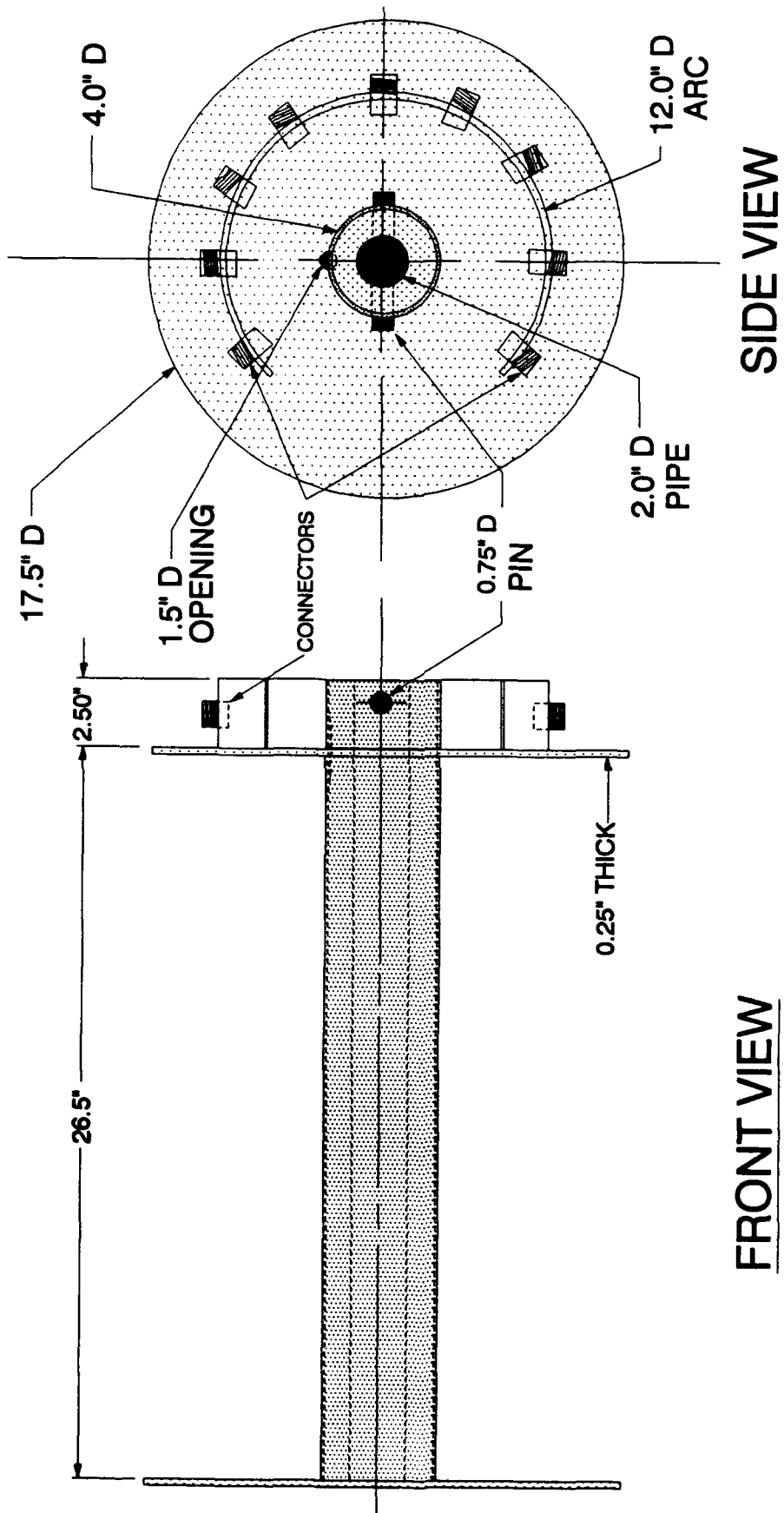
**(FOR SPOOLS #1-4,6-9)**





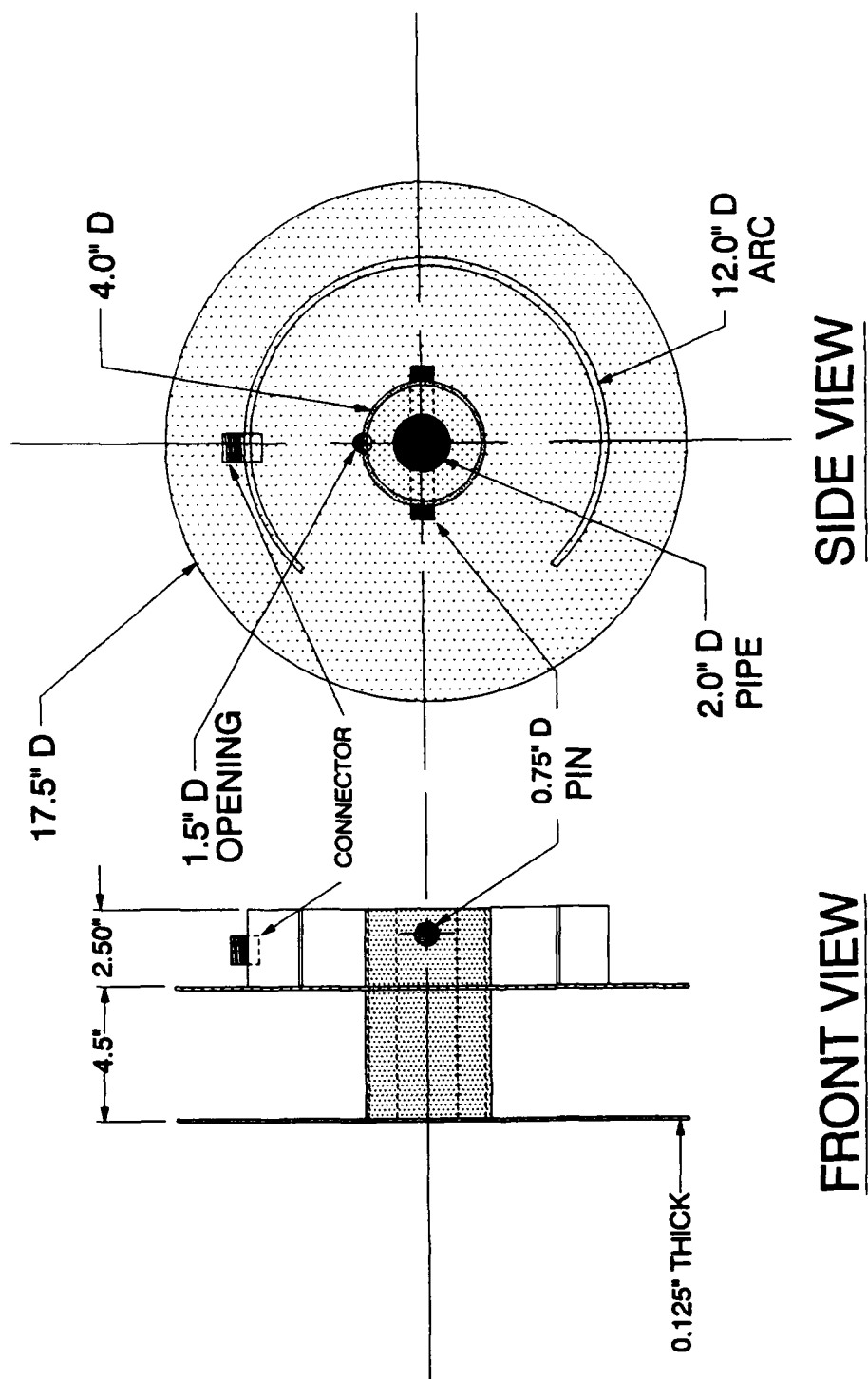
CABLES HOLDING:  
GREEN HOSE, BLACK HOSE

# SPOOL# 1



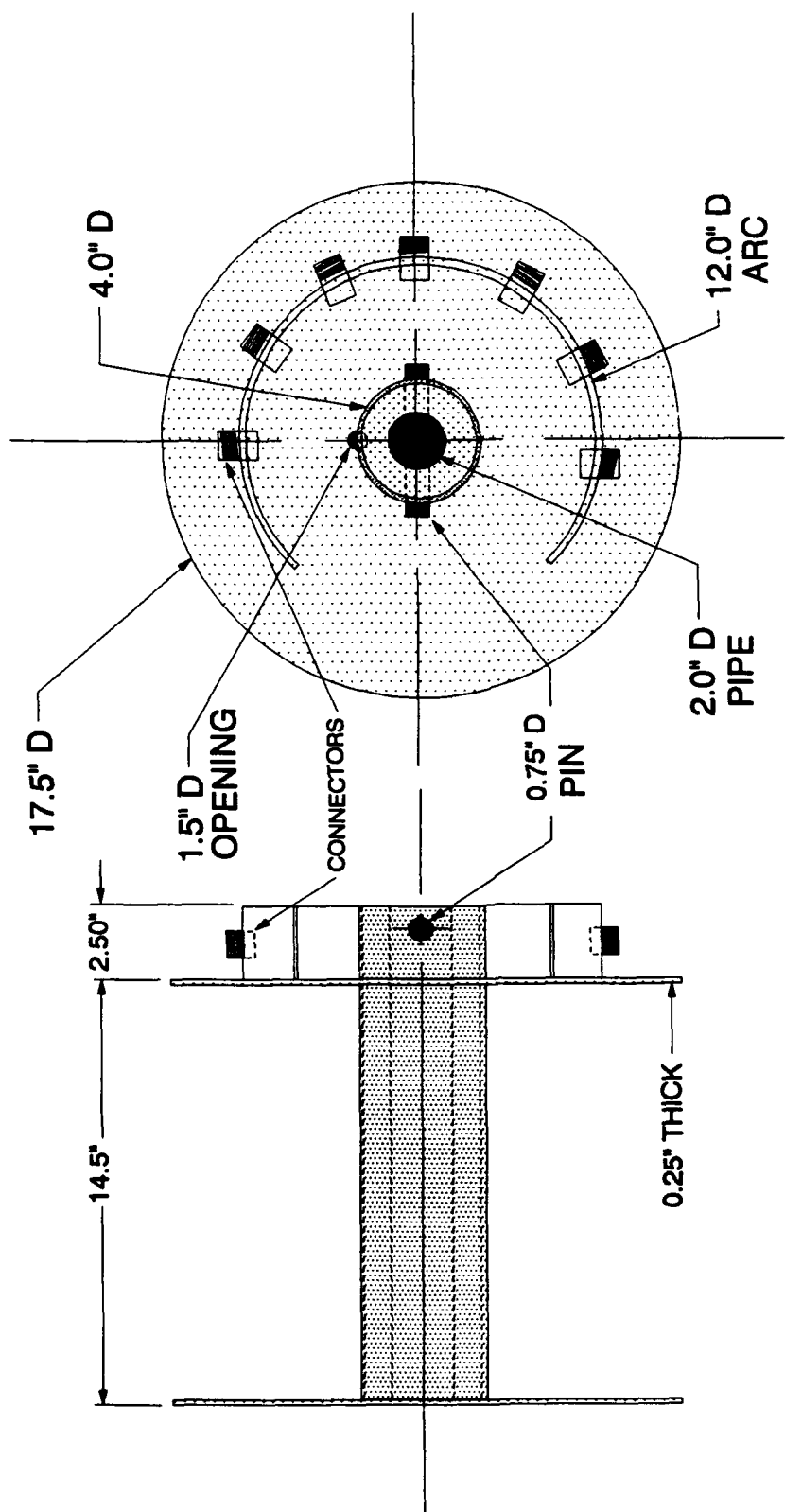
CABLES HOLDING:  
 GREEN HOSE, RED HOSE, RG 8, MAGGIE CONTROL, (3X) RG 58,  
 INTERCOM, RS 232

## SPOOL #2



CABLES HOLDING:  
LIGHT HORN CABLE

**SPOOL# 3**

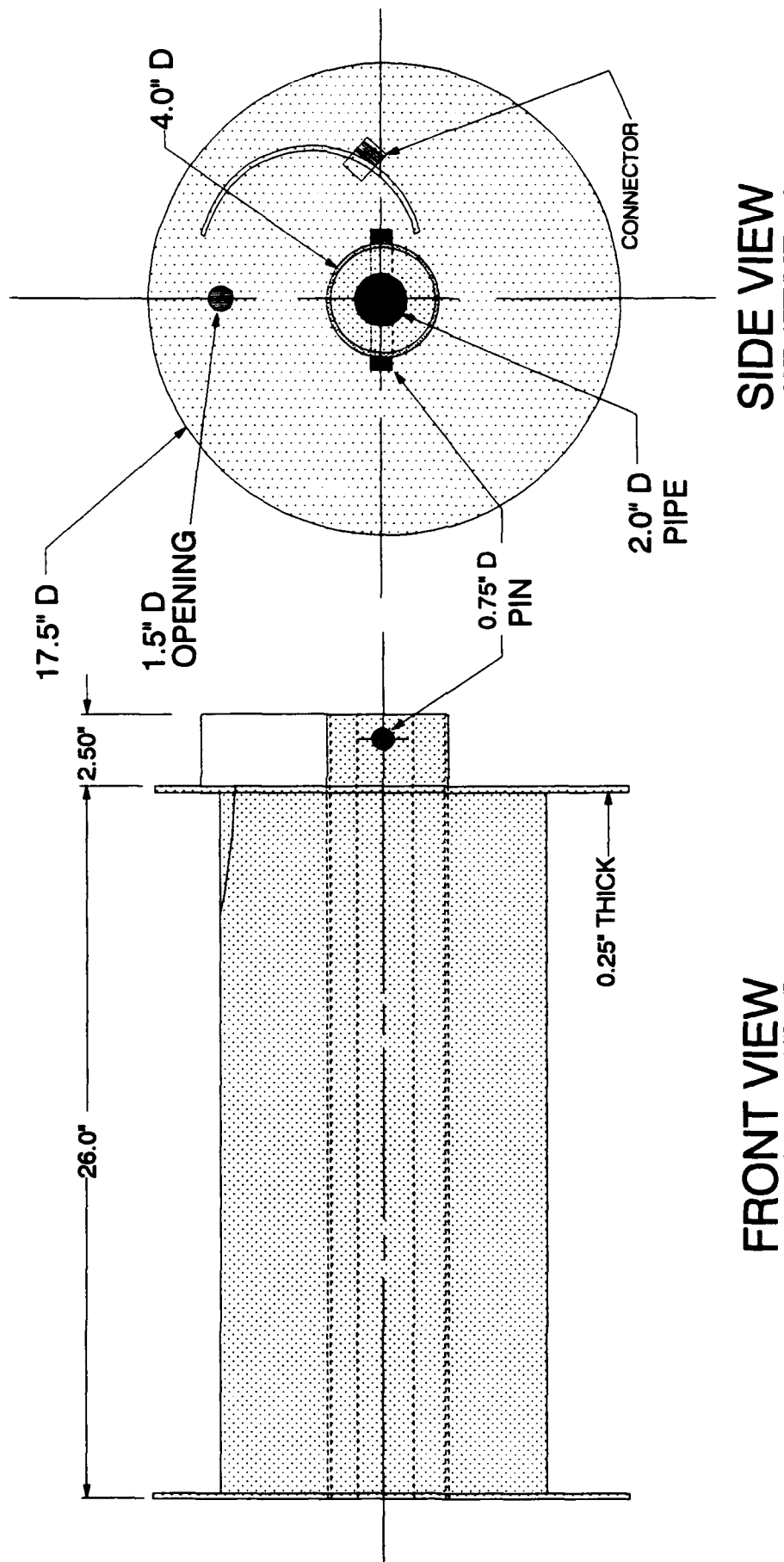


FRONT VIEW

SIDE VIEW

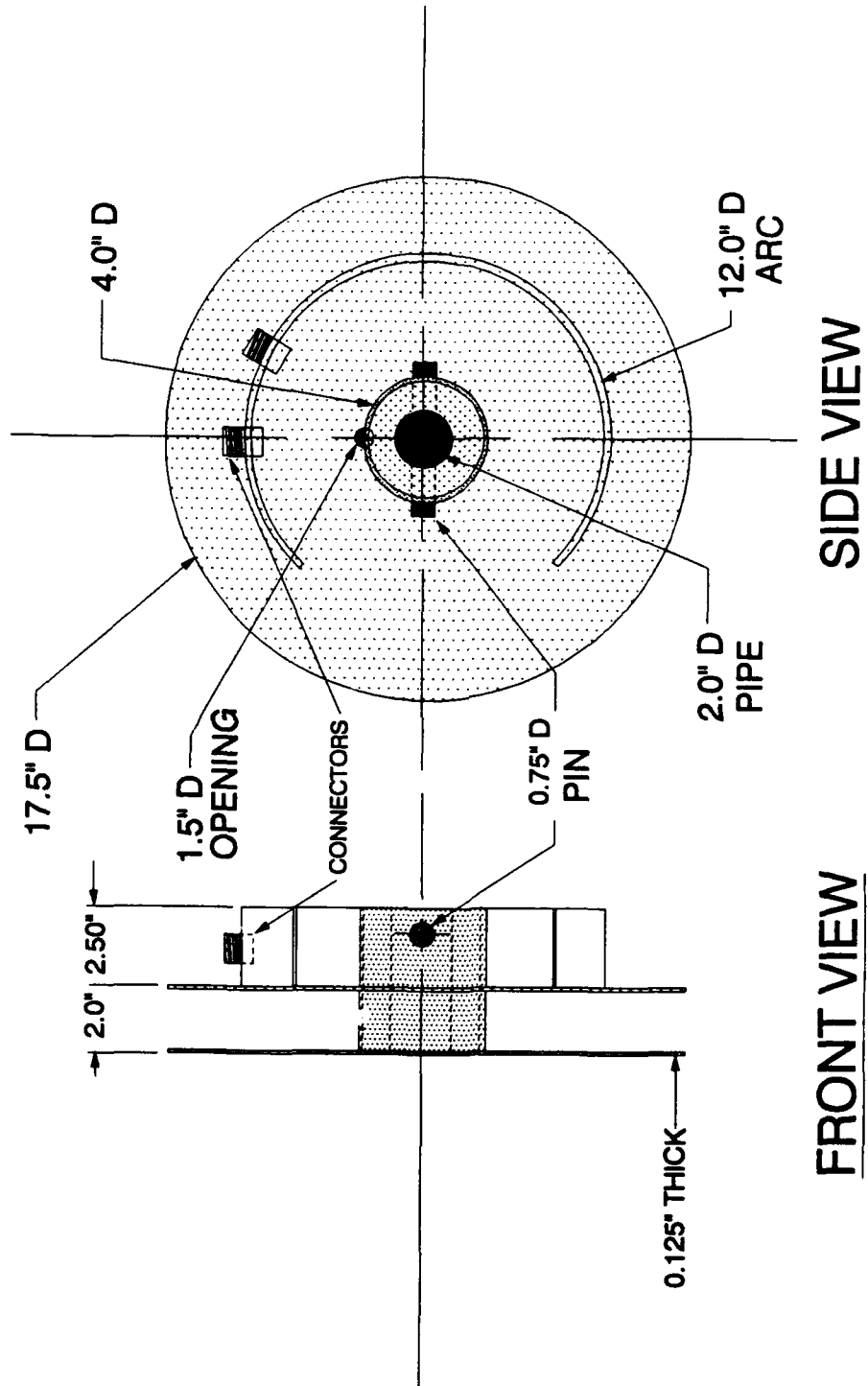
CABLES HOLDING:  
LENS CONTROL, (5) RG 59, INTERCOM

SPOOL# 4



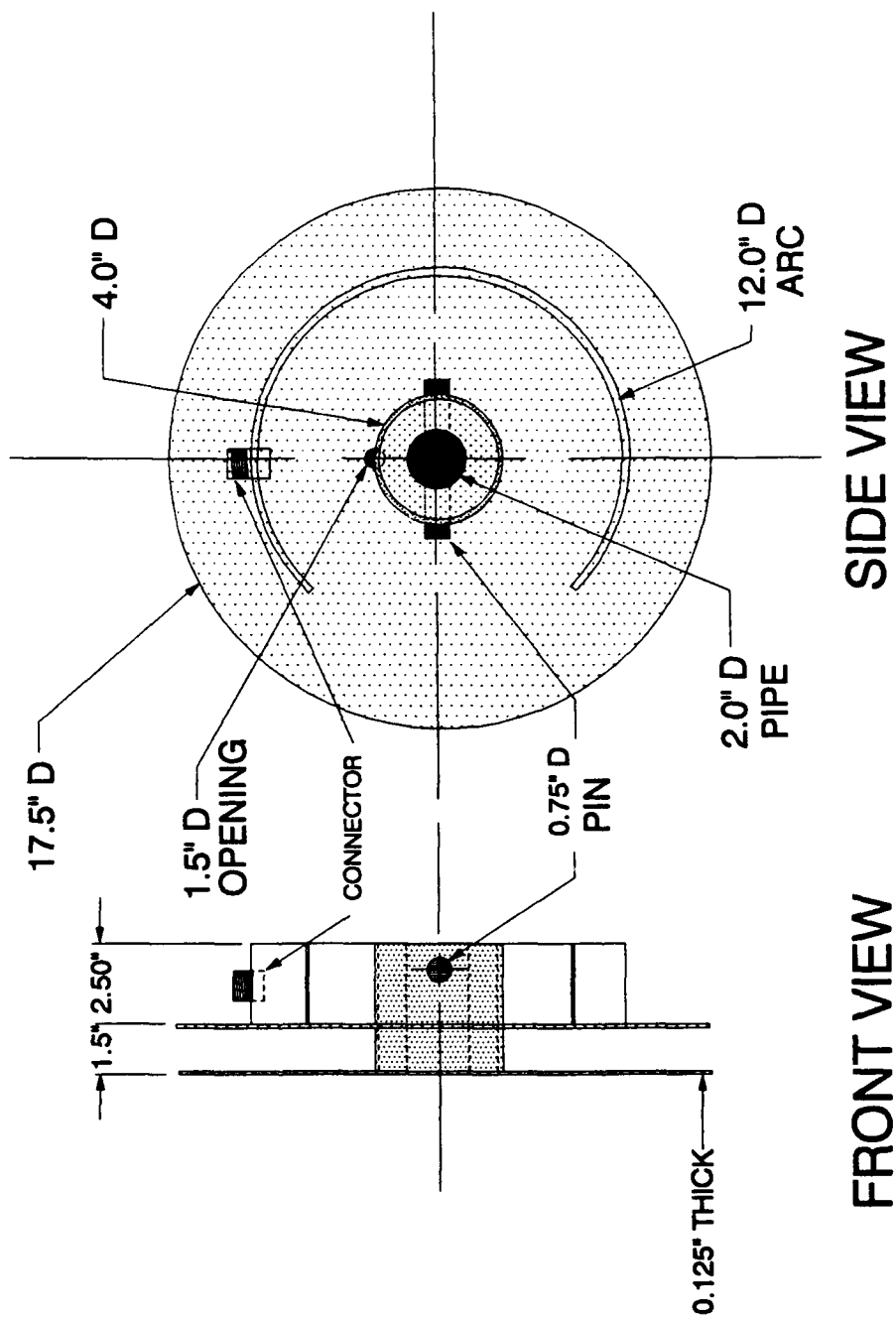
CABLES HOLDING:  
POWER CABLE

**SPOOL #5**



# SPOOL #6

CABLES HOLDING:  
 GUARD BOX, INTERCOM



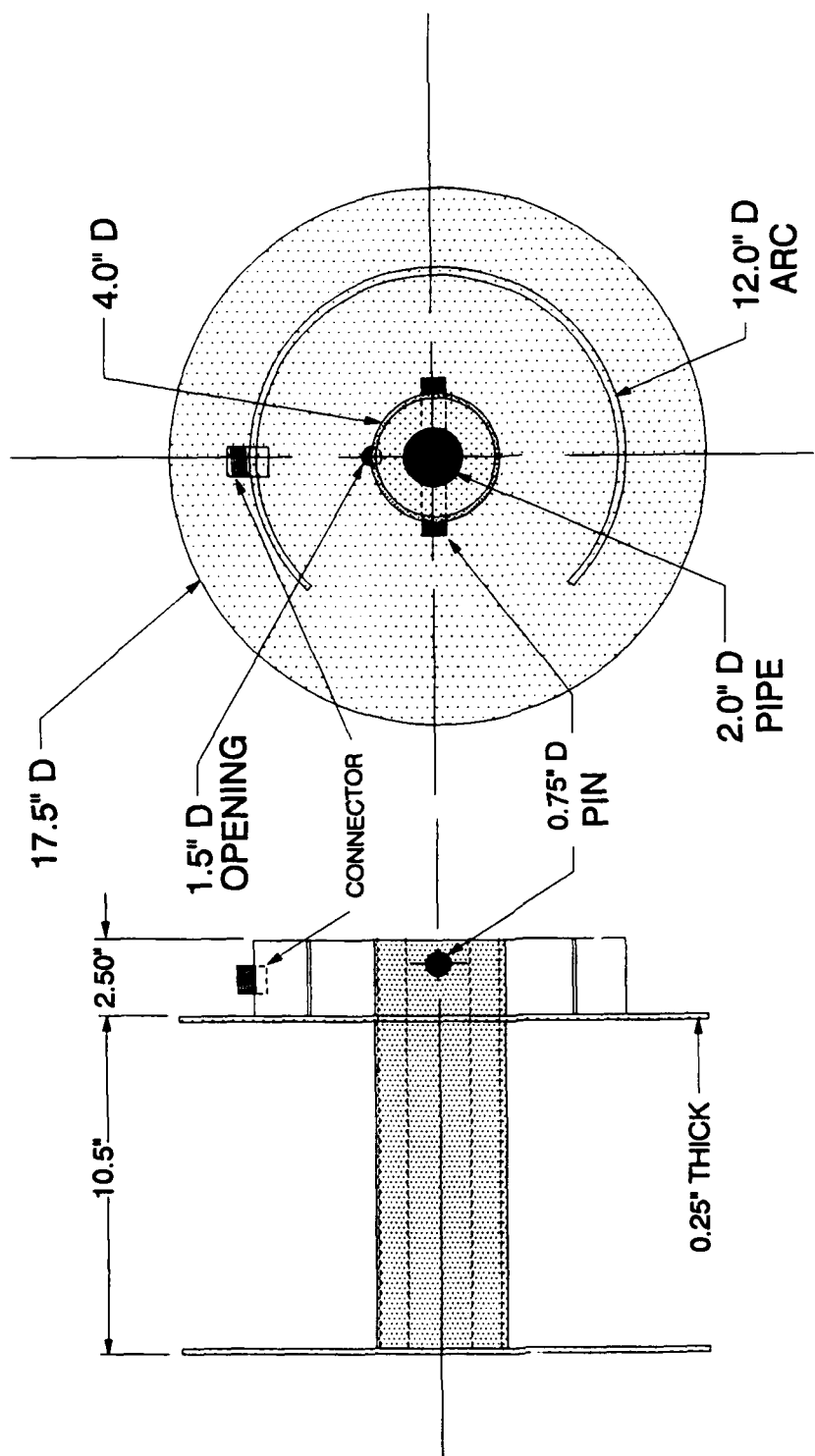
# SPOOL #7,8

CABLES HOLDING (#7):

PHONE

CABLES HOLDING (#8):

INTERCOM



FRONT VIEW

SIDE VIEW

**SPOOL #9**

CABLES HOLDING:  
ICACOM



## PROJECT MATERIALS

<u>MATERIAL</u>	<u>DESCRIPTION</u>	<u>EST. COST</u>
PLATE ALUMINUM	22.5 SQ FT, 0.25" THICK	\$200
PLATE ALUMINUM	32.0 SQ FT, 0.50" THICK	\$60
SHEET ALUMINUM	18.0 SQ FT, 0.125" THICK	\$60
SHEET ALUMINUM	4.0 SQ FT, 0.125" THICK	\$15
BEARINGS	1-7/16" SHAFT DIA, 5" MTG CENTER	\$10
STEEL PIPE	75", 2.0" DIA	\$475
ALUMINUM PIPE	4" CONDUIT	\$15
MISCELLANEOUS	HARDWARE	\$50
TOTAL ESTIMATED PROCUREMENT COSTS:		<u>\$975.00</u>

**PROJECT MANHOURS**

**EJ3**

**EL3**

**DESIGN:**

**96**

**FABRICATION:**

**8**

**48**

**INSTALLATION/CHECK OUT:**

**8**

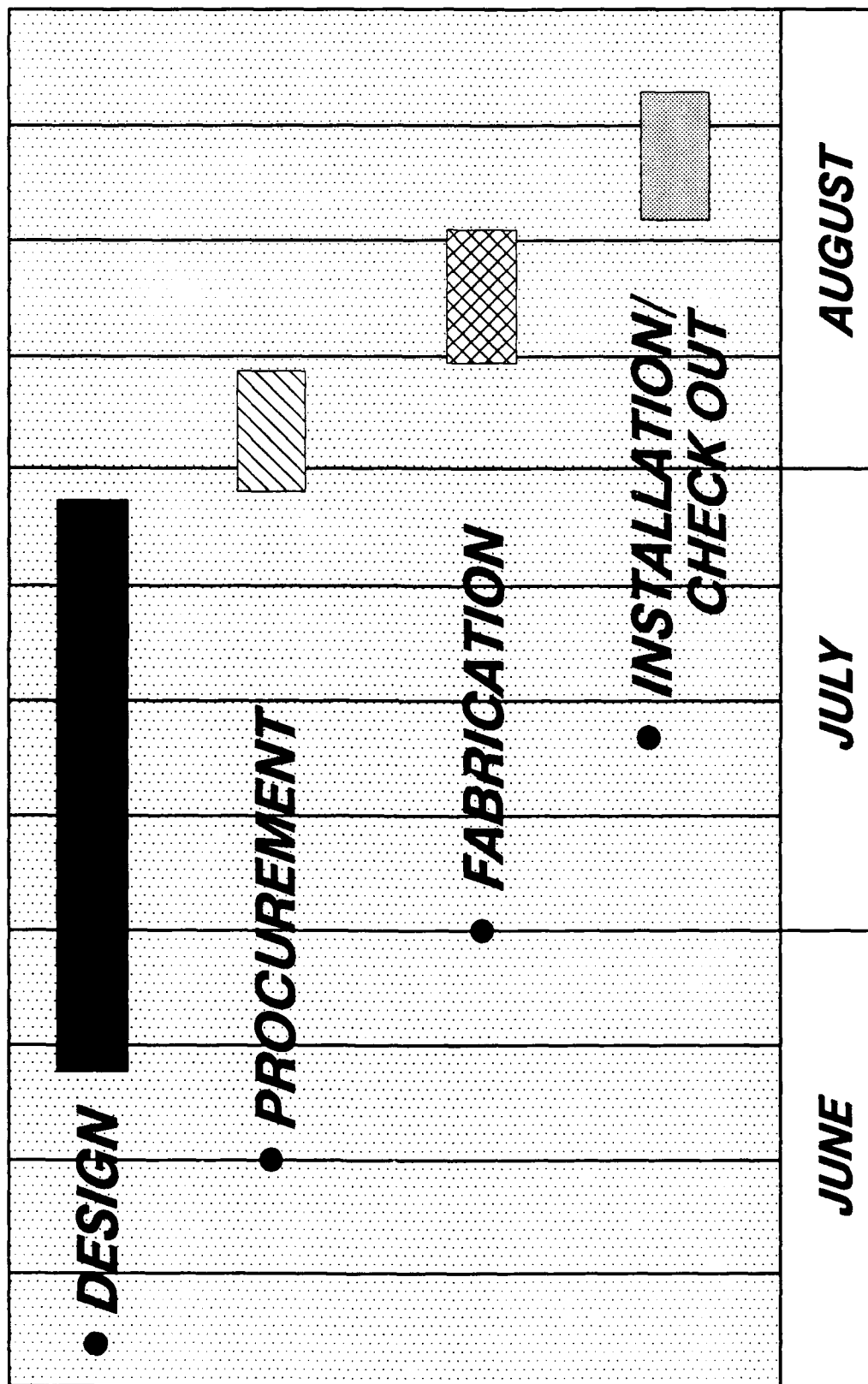
**16**

**112**

**64**

**TOTAL MANHOURS: 176 HRS**

# CABLE SPOOLING PROJECT SCHEDULE



Menu-Driven Graphical Control  
Input Software

Genetta Gibson

Mentor  
Stephen Powell

Arnold Engineering Development  
Center

August 1991

## MENU-DRIVEN GRAPHICAL CONTROL INPUT SOFTWARE

Genetta Gibson

### **Abstract**

A menu-driven program to create or modify an input file for the Graphic Interactive Analysis Techniques (GIANT) program was created. GIANT is a graphics program used to plot steady-state data for turbine engine testing on a graphics terminal or a graphics workstation. GIANT Initialize (GIANT Init) defines the parameters to be plotted by the GIANT program. The source code was written in the FORTRAN language with the use of the Plot 10 graphics library subroutines to perform input and output of the GIANT Init information. These graphics routines allow for a more user-friendly program than using standard FORTRAN code but still maintain a desired portability of the source code among the various computer systems. The program, which is called MIRAGE (Modify Input Record to Access Giant Effectively), was developed on the Apollo workstation, which runs under the Aegis operating system. MIRAGE provides on-line help for the user and input validation to help prevent GIANT from aborting during execution.

## **Acknowledgements**

The author would like to thank Stephen Powell for his time in the design of the program and in teaching the fundamentals of the FORTRAN language, Randy Sloan for his help with the operation of the Apollo workstation, Bill Lodholz for his help with the oral presentation of the program, and the EJ5 personnel for all the suggestions and guidance to make this apprenticeship program a successful learning experience.

## **General Description**

GIANT Init contains a large amount of data that is column oriented and line specific. If one input is not in the correct column, is not located on the correct line, or is of the wrong data type, an error is given. Locating the erroneous input can be a tedious task. MIRAGE uses menus to prompt for the data, provides error checks, and places the data in the correct columns and on the correct lines, which helps prevent the graphics program from aborting during execution. MIRAGE may be used to create a new file or modify an existing GIANT Init file.

Chart 1 illustrates the overview concept of MIRAGE and its relation to CIANT. Using the old process, the engineer interacted directly with the GIANT Init input file by using the editors available on the various computer systems. GIANT

then utilized the plot definitions, created the plot, and returned the plot to the engineer. With MIRAGE, the engineer enters the plot definitions into MIRAGE, which in turn writes the definitions to the GIANT Init file while providing on-line help and error checks. As before, GIANT then utilizes the definitions and creates a plot that is returned to the engineer.

At the present moment, only the GIANT Init file may be created or modified through the use of MIRAGE, but a future plan is to expand the program so that more graphical control input files may be created or modified.

#### **Detailed Description**

GIANT Init contains three major inputs -- file control, plot description, and background data. By referring to Chart 2, one can see the different subroutines as they are called for the entering of these inputs. The Graphical Input Files Control (Screen 1) allows the user to either enter into or exit out of MIRAGE. If the user chooses to enter into the program, the Modification Selection Menu (Screen 2) appears, allowing the user to either create a new GIANT Init file or modify an existing one. When the modify option is chosen, the Input Data subroutine is called to read the GIANT Init information into variables; otherwise, when a new file is created, the variables are initialized with the default values. From there, the GIANT Init Inputs menu (Screen 3)

appears, allowing the user to select the three major inputs. If File Control is selected, a menu appears (Screen 4) that allows for the creating, modifying, or deleting of a file control unit. When the create or modify options are chosen, Search File Inputs (Screen 5) is called for the entering of the file control inputs. If the Plot Description option is chosen from the GIANT Init Inputs (Screen 3) screen, the options to create, modify, or delete may be chosen (Screen 6). The Parameter Inputs screen (Screen 7), which allows for the entering of the inputs, will appear if the modify or create options are selected. Four of these inputs require an additional screen (Screens 8-11) for the entering of the actual inputs. The Background Data Inputs menu (Screen 12) may also be called from GIANT Init Inputs (Screen 3), and the screen allows the user to create, modify, or delete a background data input. When the create or modify option is selected, the Background Data Set Inputs menu (Screen 13), which allows for the entering of the inputs, appears. The symbol and value inputs require additional screens (Screens 14 and 15) for the actual entering of inputs.

All screens are equipped with an option to enter a "?" for help (except 2, 10, and 14, for which there is no need for help). In addition, the options may be chosen by either the number or the capital letter of the option. When most options are selected, a prompt appears asking for the particular input, as shown in Chart 3. If the input entered



is valid, it will appear beside the option, as shown in Chart 4. If the input is invalid, an error message will appear, and the input will not appear beside the option. (The default value would appear instead.) Some inputs are toggle options which alternate between "YES" and "NO" depending on which is chosen. Chart 5 shows how a plot description input might appear through the use of MIRAGE, and Chart 6 displays a GIANT Init file with the information in Chart 5 highlighted. As can be seen, the data file appears in a very disorderly fashion that is extremely difficult to read and interpret.

MIRAGE, which was developed on the Apollo workstation and written in standard FORTRAN, may be transported to other computer systems (e.g. Amdahl and VAX) with only minor difficulties.

### **Summary**

An interactive menu-driven program (MIRAGE), which allows the engineer to create a new graphical control input file or modify an existing file, was developed. The software development was performed on the Apollo workstation. The program was written in the FORTRAN 77 language using Plot 10 graphics library subroutines for portability of source code to the various computer systems.

Review sessions of the software with engineering personnel were conducted. Their responses showed that the

program would be very useful to the occasional users of GIANT and helpful to everyone except the "GIANT experts."

### **Observations**

Through creating a menu-driven program to build the GIANT Init file, the author was able to learn the fundamentals of the FORTRAN programming language and the Apollo workstation, gain knowledge of the computer programming profession through firsthand experience, and gain a better understanding of the interaction between programmers and engineers.

The author considers the High School Apprenticeship Program to be a success in the field of technology and an excellent learning experience for any teenager who expresses an interest in a technical field. The author would recommend the program to any interested student and plans to apply next year.

### **Bibliography**

Merchant, Michael J. **FORTRAN 77 Language and Style.**  
California: Wadsworth Publishing Company, 1981.

**Plot 10 Graphics Software Users Manual.** Tektronix,  
Inc., 1971.

**Programming in VAX FORTRAN.** Digital Equipment  
Corporation, 1984.

Smith, R.W. **GIANT Users Guide.** July 1984.

## GRAPHICAL INPUT FILES CONTROL

1. Giant init input file
2. Exit

(enter ? for help)

Screen 1

---

## MODIFICATION SELECTION MENU

1. Create a new input file
2. Modify an existing input file
3. Exit to main menu

Screen 2

---

## GIANT INIT INPUTS

1. File control inputs
2. Plot description inputs
3. Background data inputs
4. Exit and save inputs
5. Quit without saving inputs

(enter ? for help)

Screen 3

## FILE CONTROL INPUTS

1. Add a new search file unit
  2. Change an existing search file unit
  3. Delete an existing search file unit
  4. Match project number on data point and index records YES
  5. Exit and save file control inputs
  6. Quit without saving file control inputs
- (enter ? for help)

Screen 4

---

## SEARCH FILE UNIT INPUTS

1. Unit number of file XX
  2. Pre-symbol for parameter names X
  3. Replacement symbol for parameter names X
  4. First location for name search XXXX
  5. Last location for name search XXXX
  6. Alpha name information listed YES
  7. Exit and save search file unit inputs
  8. Quit without saving search file unit inputs
- (enter ? for help)

Screen 5

## PLOT DESCRIPTION INPUTS

- |  | NUMBER OF<br>PLOTS DEFINED |
|--|----------------------------|
| 1. Add a new plot description              |                            |
| 2. Change an existing plot description     | XX                         |
| 3. Delete an existing plot description     |                            |
| 4. Exit and save plot description inputs   |                            |
| 5. Quit without saving file control inputs |                            |

(enter ? for help)

Screen 6

---

## NAME AND PARAMETER INPUTS

- |  |            |
|--|------------|
| 1. X parameter name                      | XXXXXXXXXX |
| 2. Y parameter name                      | XXXXXXXXXX |
| 3. Minimum x-axis value                  | XXXXXX     |
| 4. mAximum x-axis value                  | XXXXXX     |
| 5. miNimum y-axis value                  | XXXXXX     |
| 6. maxIum y-axis value                   | XXXXXX     |
| 7. File item locations specified         | YES        |
| 8. Line control used                     | YES        |
| 9. Grid lines                            | YES        |
| 0. Background references                 | X X X      |
| &. Exit and save parameter inputs        |            |
| \$. Quit without saving parameter inputs |            |

(enter ? for help)

Screen 7

## FILE ITEM LOCATIONS

	DATA FILE	X-ITEM	Y-ITEM
1.	XX	XXX	XXX
2.	XX	XXX	XXX
3.	XX	XXX	XXX
4.	XX	XXX	XXX
5.	XX	XXX	XXX

6. Exit and save file item locations
7. Quit without saving file item locations

(enter ? for help)

Screen 8

---

## LINE CONTROL INPUTS

	DATA FILE	LINE CONTROL
1.	XX	X
2.	XX	X
3.	XX	X
4.	XX	X
5.	XX	X

6. Exit and save line control inputs
7. Quit without saving line control inputs

(enter ? for help)

Screen 9

## GRID LINE INPUTS

1. number of grid lines drawn  
parallel to the X-axis XX
2. number of grid lines drawn  
parallel to the Y-axis XX
3. Exit and save grid line inputs
4. Quit without saving grid line inputs

Screen 10

---

## BACKGROUND REFERENCE INPUTS

### DEFINED BACKGROUND REFERENCE SET NUMBERS

1. background reference set number 1 XX
  2. background reference set number 2 XX
  3. background reference set number 3 XX
  4. Exit and save background reference inputs
  5. Quit without saving background reference inputs
- (enter ? for help)

Screen 11

## BACKGROUND DATA INPUTS

1. Add or modify a background data set
2. Delete an existing background data set
3. Exit and save background data inputs
4. Quit without saving background data inputs

(enter ? for help)

Screen 12

---

## BACKGROUND DATA SET INPUTS

Background Data Set Number: XX

1. Symbol for background data values
2. Incremental value for background symbol
3. Define background data values
4. Exit and save background data set inputs
5. Quit without saving background data set inputs

(enter ? for help)

Screen 13



## BACKGROUND DATA SYMBOL INPUTS

Current Code for Background Symbol: XX

0. No symbols on background data
1. select symbol "O"
2. select symbol "X"
3. select symbol "Triangle"
4. select symbol "Rectangle"
6. select symbol "Diamond"
7. Values will / will not be joined by line segments
8. Exit and save background data symbol inputs
9. Quit without saving background data symbol inputs

Screen 14

---

## BACKGROUND DATA VALUES

Data Set Number: XX

X	Y	X	Y	X	Y
1.		16.		31.	
2.		17.		32.	
3.		18.		33.	
4.		19.		34.	
5.		20.		35.	
6.		21.		36.	
7.		22.		37.	
8.		23.		38.	
9.		24.		39.	
10.		25.		40.	
11.		26.		41.	
12.		27.		42.	
13.		28.		43.	
14.		29.		44.	
15.		30.		45.	

- |                           |                                    |
|---------------------------|------------------------------------|
| 1. Add new points         | 6. Break connecting line segments  |
| 2. Modify existing points | 7. add Polynomial function inputs  |
| 3. Delete existing points | 8. Exit and save data values       |
| 4. add xmiN / ymiN values | 9. Quit without saving data values |
| 5. add xmaX / ymaX values | (enter ? for help)                 |

Screen 15

Chart 1

# Process Overview

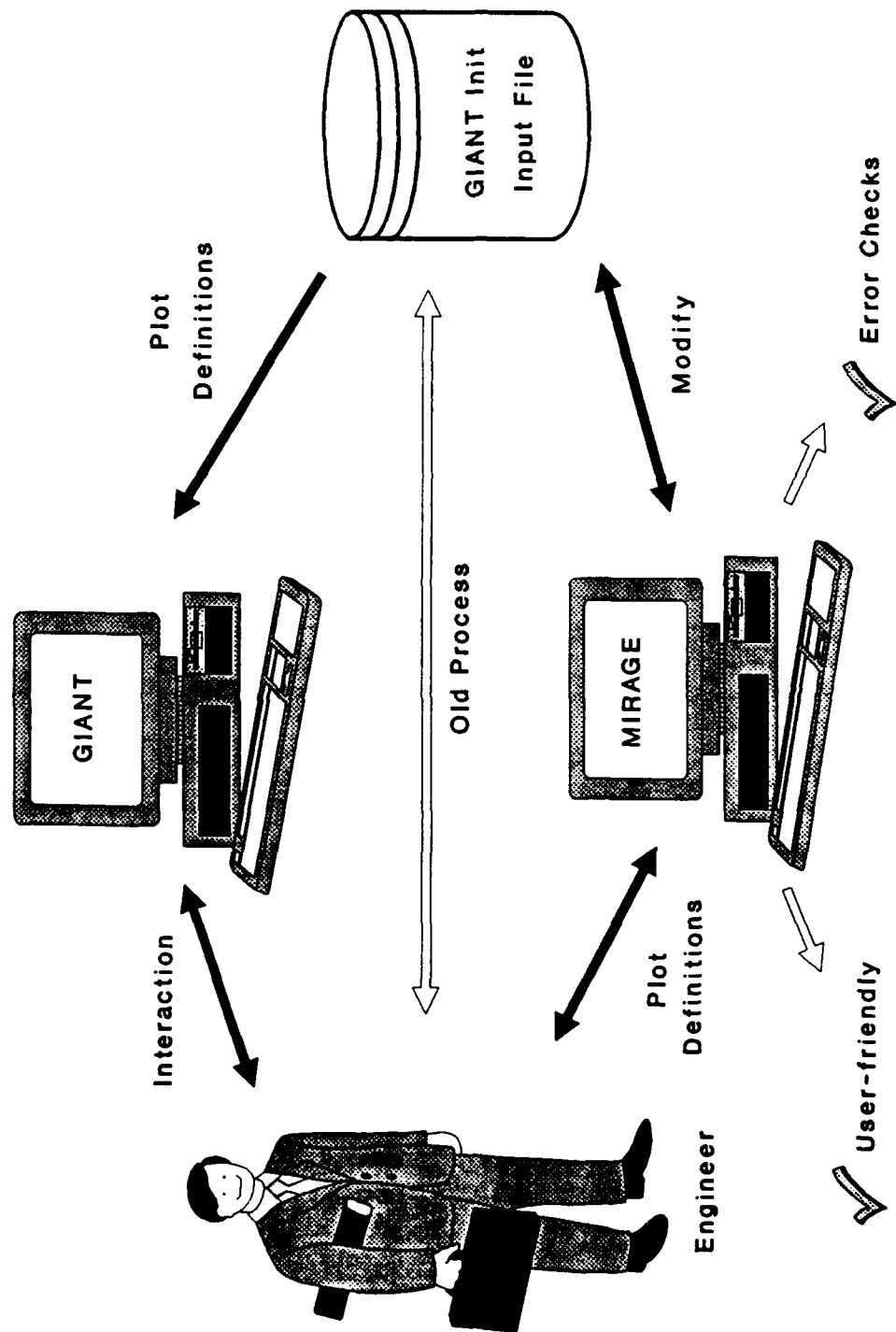
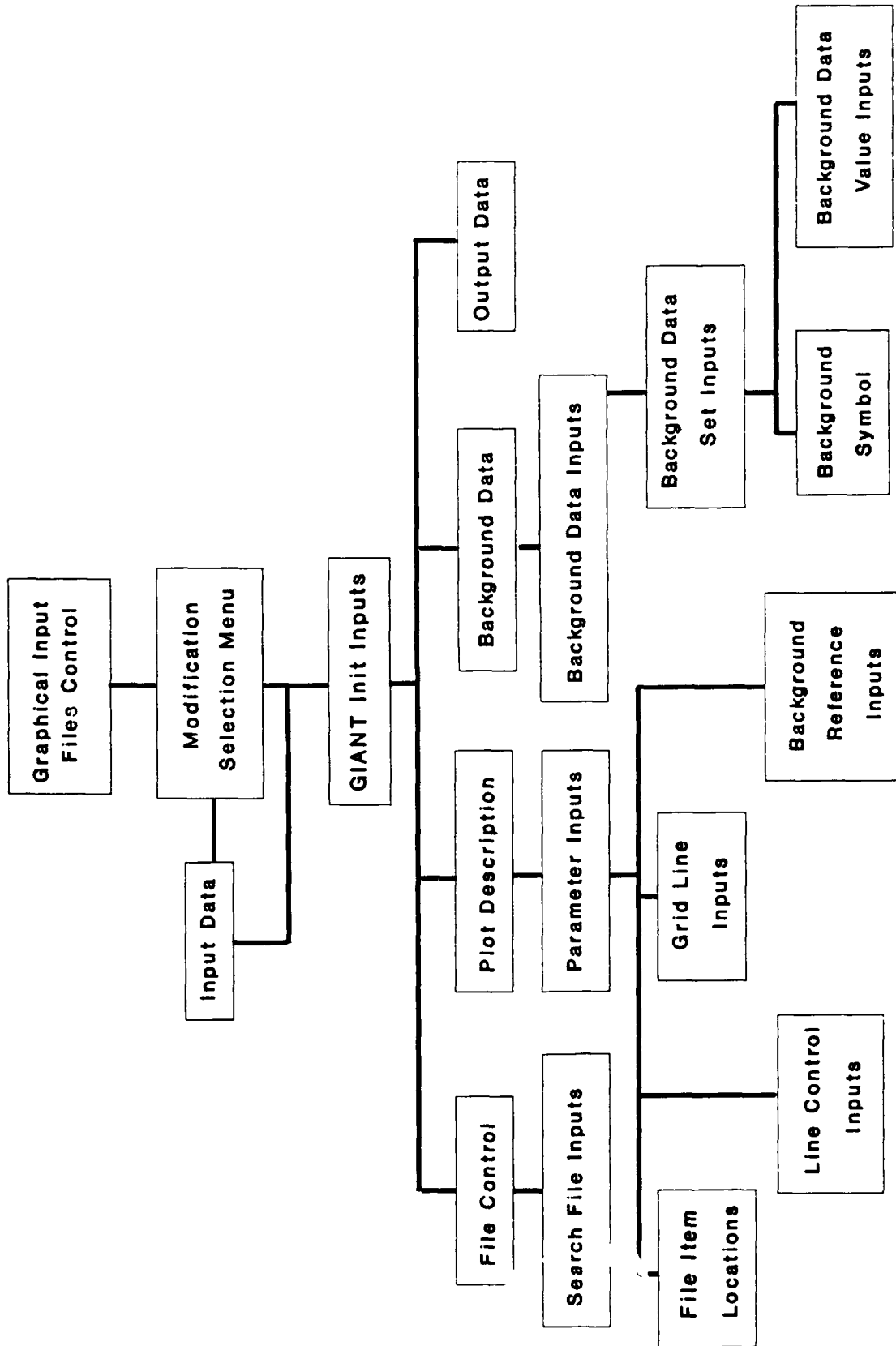


Chart 2

# Hierarchical Model



## NAME AND PARAMETER INPUTS

Chart 3

1. X parameter name
2. Y parameter name
3. Minimum x-axis value
4. mAximum x-axis value
5. miNimum y-axis value
6. maxIum y-axis value
7. File item locations specified
8. Line control used
9. Grid lines
0. Background references
- &. Exit and save parameter inputs
- \$. Quit without saving parameter inputs  
(enter ? for help)

ENTER THE X PARAMETER NAME: FAR413

## NAME AND PARAMETER INPUTS

Chart 4

FAR413

1. X parameter name
  2. Y parameter name
  3. Minimum x-axis value
  4. mAximum x-axis value
  5. miNimum y-axis value
  6. maxIum y-axis value
  7. File item locations specified
  8. Line control used
  9. Grid lines
  0. Background references
  - &. Exit and save parameter inputs
  - \$. Quit without saving parameter inputs
- (enter ? for help)

# NAME AND PARAMETER INPUTS

Chart 5

1. X parameter name	FAR413
2. Y parameter name	EFFX
3. Minimum x-axis value	100
4. mAXimum x-axis value	1000
5. miNimum y-axis value	600
6. maxIum y-axis value	6000
7. File item locations specified	YES
8. Line control used	NO
9. Grid lines	YES
0. Background references	2 11
&. Exit and save parameter inputs	
\$. Quit without saving parameter inputs	
(enter ? for help)	

# GIANT Init Input File

Chart 6

```

41& 567 1121# 41 785 3421 86T 230 897H 41& 376 234% 41& 986 1459      IIII 4
GIANT INITIALIZE
FAR413 ETACT 0.03 0.05 96.0 101.0      G10 5
BACKGROUND DATA 2 9
FARETA SA5XRM 0.02 0.06 3.0 5.0      L C G 4 10
BACKGROUND DATA 3
[REDACTED]      G10 10
FAR405 SA5RT2 0.20 6.0      G10 15
BACKGROUND DATA 5
ALT X MACHX 0.060000 0.0 3.0 FILEITEM      L G 6 10
869 870 869 870 869 870 869 870 869 870
BACKGROUND DATA 6
FAR413 ETACT      G15 8
BACKGROUND DATA 5

2 3
.02500099.92000 .02597799.99788 .026953300.0860 .026930200.2644 .028907200.2325
.0254760.459864 .0239874.982355 .0984535.889353 .032824200.4470 .0934529.563849
.8933456.234006 .0234544.993443 .6733453.093453 .02857492.90872 .03445743.98435
-999999.0

6 -3
.03456578.78555 .09486758.89486 .0984785.874589 .043687846.9804 .08456584.43546
.034867887.8948 .08789475.89458 .088746356.7834 .09456328.90846 .08346545.90038
.09787586.78495 -999999.0
    
```

**REMOTE ACCESS TO INSTRUMENT BOOK INFORMATION**

**Kenneth Harwell**

**Mentor**

**Harry Clark**

**AEDC**

**8-14-91**



## **ABSTRACT**

The purpose of the Remote Access to Instrument Book Information Project was to develop a computer generated instrument book for the instrument technicians at the Data Input Center (DIC). This database will replace the old bulky instrument book. The technicians needed a way to refer to the information about the parameters and be able to make comments about a certain parameter of choice. This display of information will improve the method of communication between the Data Input Center (DIC) and the EAF building. Updating the computerized instrument book will be quicker and more efficient. This improved instrument book used by the technicians will help in the time of preparation before testing which leads to improved testing at Arnold Engineering and Development Center(AEDC).

Appendix A contains example program output screens and selected programs.

During the Air Force Apprenticeship program, Harry Clark was my mentor. I would like to thank him for giving me the opportunity to work with his research group and showing me the rewards of engineering at AEDC. This summer I worked with many engineers and technicians in the EJ-2 work area. I would also like to thank them for treating me like an equal and helping me fit in at AEDC. Research and Development Laboratories (RDL) and the Air Force gave me and twelve others the opportunity to work at AEDC this summer. I would like to thank them for their support.

The apprenticeship project was to develop a program for the Remote Access to Instrument Book Information which will improve the communication between the technicians in the field and the engineers in the office. The main communication link is between the Data Input Center (DIC) and the EAF building. This communication will make updating and using the instrument book efficient.

My approach in generating the Remote Access to Instrument Book Information program was to make the program user friendly and to aid in finding the appropriate parameter and information quickly and easily. First I needed to learn DBASE III+ to actually write the program. I also learned wordprocessing (Word Star) to update the program. After learning and becoming familiar with the two languages, I set up a simple database of 50 records to make programming easier to manage before using the large instrument book database. After becoming familiar with DBASE III+ and Word Star, I started working with the instrument book database. Next a list of nine parameter names was generated on the screen. A highlighted bar covers the parameter that you can choose by pressing the enter key. By pressing the down arrow key the highlighted bar will move to the next parameter name. After completing the first nine names the screen will show the next nine and repeat the process. Also if the up arrow key is entered the highlighted bar will move to the previous parameter name on the list of nine. If the page down key is pressed the next nine parameters will be displayed on the screen and the page up key will display the previous nine parameter names. If a letter or number is entered the program will look for the corresponding first character of the name and display it on the screen. When the highlighted bar is over a certain parameter name then it can be selected by pressing return to show certain essential information

about the parameter to the technician. This information is what the technicians work with and use daily to set up equipment. If the information is incorrect or they have questions about the information there is a comment line at the bottom of the screen where they can input comments about the parameter. The program is the main communication device between the DIC and the EAF building. This program is beneficial to both technicians and engineers in communication of problems in the field or in the instrument book, but these problems can only be changed by the engineers in the EAF building. The Remote Access to Instrument Book Information Program provides a better and more effective communication between the DIC and the EAF building.

This summer at AEDC exposed me to the process of rocket testing. I observed three hot firing rocket tests at the J5 test cell. Two of the tests were Minuteman Stage 3 and the other was a Minuteman Stage 2. During my time at AEDC I was able to see the importance of rocket testing to keep our defense systems in perfect firing condition. The rocket testing at AEDC is important in maintaining our country as a front runner in technology and development of rockets. I hope this computer program will help in the future testing of rocket motors at AEDC.

## **APPENDIX A**

**MENU**

**GENERAL**

**NON-GENERAL**

**SECRET**

**MENU**

**GENERAL**

**NON-GENERAL**

**PARAMETERS**

**PARAMETERS**

**EQA  
FX-1  
FX-2  
FX-3  
FYA-1  
FYA-2  
FYA-3  
FYP-1  
FYP-2**

Use ENTER to select item  
Use UP and DOWN ARROW keys to move  
Use F1 key to quit  
Use HOME to return to beginning

**MENU**

**GENERAL**

**NON-GENERAL**

**PARAMETERS**

**PARAMETERS**

**EQA  
FX-1  
FX-2  
FX-3  
FYA-1  
FYA-2  
FYA-3  
FYP-1  
FYP-2**

Use ENTER to select item  
Use UP and DOWN ARROW keys to move  
Use F1 key to quit  
Use HOME to return to beginning

**MENU**

**GENERAL**

**NON-GENERAL**

**PARAMETERS**

---

**PARAMETERS**

**FYA-2  
FYA-3  
FYF-1  
FYF-2  
FYF-3  
PA-1  
PA-2  
PA-6  
PDTOT-1**

---

Use ENTER to select item  
Use UP and DOWN ARROW keys to move  
Use F1 key to quit  
Use HOME to return to beginning

**MENU**

**GENERAL**

**NON-GENERAL**

**PARAMETERS**

---

**PARAMETERS**

**FYA-2  
FYA-3  
FYF-1  
FYF-2  
FYF-3  
PA-1  
PA-2  
PA-6  
PDTOT-1**

**NONE OF THIS TYPE : ♦**

---

**Use ENTER to select item  
Use UP and DOWN ARROW keys to move  
Use F1 key to quit  
Use HOME to return to beginning**



MENU

GENERAL

NON-GENERAL

PARAMETERS

PARAMETERS

PARAMETER EFS-1

INTERFACE PB1, A-21

JB 550, TB-19, TERM. 61

TRJ547-1

TRJ547-2

EFS-1

EFS-2

EFS-3

EFS-4

EFS-5

EFS-6

EOM-1

CHANNEL INFORMATION

CHANNEL A-075 CONDITIONER MA-OC-OG-K EXCITATION N/A

FILTER

GAIN

5301.1

500

200

5301.2

200

PATCHING

INPUT D-15/16

5301.3

1883

200

OUTPUT T-28

5301.4

500

5000

5301.1

5301.2

5301.3

5301.4

RECORDERS

DACS

MPX-B

OSC-B

PDDS

AUX RECORDERS INPUT PATCHES

HARDW

W-23

Y-08

CC-08

COMMENT :

```

* MENU.PRG
CLEAR
SET STATUS OFF
SET INTENSITY ON
SET TALK OFF
SET DEVICE TO SCREEN
CLEAR ALL
SET CONSOLE OFF
SET COLOR TO N/W
CLEAR
X=0
Q=0
I=4
DO WHILE X<4
SET COLOR TO R/W
@ 4,0 TO 4,76 DOUBLE
@ 19,0 TO 19,76 DOUBLE
SET COLOR TO G/W
@ 0,29 TO 2,40 DOUBLE
@ 0,41 CLEAR TO 0,79
SET COLOR TO R/B
@ 1,32 SAY 'MENU '
IF Q=1
DO WHILE I=0
* INPUT TO SELECT FROM MENU
I=INKEY()
Q=1
ENDDO
SET COLOR TO R/W
@ 20,0 CLEAR TO 21,76
ENDIF
Q=1
DO CASE
* INKEY CASE
CASE I=4
* RIGHT ARROW KEY IS 4
X=X+1
SET COLOR TO B/G
I=0
DO CASE
* MOVING HIGHLIGHTED AREA
CASE X=1
SET COLOR TO N/W
@ 1,50 SAY '
SET COLOR TO B/G
@ 3,1 SAY 'GENERAL'
SET COLOR TO G/W
@ 3,30 SAY 'NON-GENERAL'
@ 3,60 SAY 'PARAMETERS'
CASE X=2
SET COLOR TO N/W
@ 1,50 SAY '
SET COLOR TO B/G
@ 3,30 SAY 'NON-GENERAL'
SET COLOR TO G/W
@ 3,1 SAY 'GENERAL'
@ 3,60 SAY 'PARAMETERS'
CASE X=3
SET COLOR TO N/W
@ 1,50 SAY '

```

```

SET COLOR TO B/G
@ 3,60 SAY 'PARAMETERS'
SET COLOR TO G/M
@ 3,1 SAY 'GENERAL'
@ 3,30 SAY 'NON-GENERAL'
X=0
ENDCASE
CASE I=13
  * RUN PROGRAM SELECTED
  DO SELECT
CASE I=28
  * QUIT
  * F1 KEY IS 28
  QUIT
OTHERWISE
  * BAD COMMAND ERROR TRAP
  SET COLOR TO G/R
  @ 20,30 SAY 'ENTER ANOTHER COMMAND'
  SET COLOR TO G/M
  @ 21,20 SAY 'USE -> KEY TO MOVE , ENTER TO SELECT'
  SET COLOR TO G/M
ENDCASE
I=0
ENDDO
RETURN
* END MENU
* SELECT.PRG
DO CASE
  CASE I=13
    IF X=0
      DO KK
    ENDIF
    IF X=1
      DO FIND
    ENDIF
    IF X=2
      DO FIND
    ENDIF
  ENDCASE
  * END SELECT

```

```

      * KK.PRG
* MAIN PROGRAM
SET COLOR TO R/N,R/B
USE BC67BOOK.DBF
SET SAFETY OFF
SET TALK OFF
SET DEVICE TO SCREEN
SET BELL OFF
* SETTING VARIABLES
SET COLOR TO W/M
@ 0,41 CLEAR TO 0,79
SET COLOR TO R/N,R/B
F=RECCOUNT()
NUM=1
Q=0
X=1
A=0
I=0
P=1
@ 5,0 CLEAR TO 18,79
DO WHILE I<>81
SET COLOR TO R/M
@ 5,0 CLEAR TO 5,79
@ 18,0 CLEAR TO 18,76
@ 5,16 CLEAR TO 18,79
SET COLOR TO G/M
@ 20,0 CLEAR TO 23,79
@ 20,20 SAY 'Use ENTER to select item'
@ 22,20 SAY 'Use F1 key to quit'
@ 21,20 SAY 'Use UP and DOWN ARROW keys to move'
@ 23,20 SAY 'Use HOME to return to beginning'
* INSTRUCTIONS TO SCREEN
SET COLOR TO R/M
IF Q=1
DO WHILE I=0
I=INKEY()
ENDDO
ELSE
I=1
ENDIF
Q=1
DO CASE
CASE I=1
* HOME
* HOME KEY IS 1
* GO TO BEGINNING OF LIST
P=1
NUM=1
DO LIST
CASE I=13
* RETURN
* ENTER KEY IS 13
* PUT OTHER INFORMATION ON SCREEN
IF NUM<>1
A=NUM-1
ELSE
A=F
ENDIF
IF A<F+1 .AND. A>=1
GOTO A

```

```

ENDIF
@ 8,20 SAY '
DO LOOK
DO CHANGE
* TO MAKE CHANGE
IF RECNO()<F
  SKIP (10)
ENDIF
CASE 1=5
  * UP
  * UP ARROW KEY IS 5
  * TO MOVE HIGHLIGHT UP ON ABOVE INFO.
DO LISTUP
CASE 1=24
  * DOWN
  * DOWN ARROW KEY IS 24
  * MOVE HIGHLIGHT TO NEXT INFO
DO LIST
CASE 1=3
  * NEXT PAGE
  * PGDN KEY IS 3
  * MOVE TO NEXT PAGE OF INFO
DO PAGE
CASE 1=28
  * QUIT
  * F1 KEY IS 28
  * RETURNS TO MAIN MENU
DO MENU
CLOSE ALL
X=10
CASE 1=18
  * PAGE UP
  * PGUP KEY IS 18
  * MOVES TO PAGE BEFORE
DO PGUP
OTHERWISE
  * LOOKS FOR KEY ENTERED
  C=NUM
DO SEARCH
  NUM=C
ENDCASE
I=0
ENDDO
RETURN
CLOSE ALL
* END OF KK.PRG

```

```

* LIST.PRG
* MOVE HIGHLIGHTED AREA DOWN TO NEXT
IF NUM<F+1
GOTO NUM
IF P<1 .OR. P>=10
P=1
ENDIF
SET COLOR TO R/W
@ 0,41 CLEAR TO 0,79
IF P=1
@ 5,0 CLEAR TO 18,79
DO BOX
SET COLOR TO R/B
@ 6,2 SAY ' PARAMETERS '
SET COLOR TO R/W
X=1
DO WHILE X<10
@ X+7,3 SAY CODE
IF .NOT. EOF()
SKIP
ENDIF
DO WHILE SUBSTR(ITEM,6,1) <> '1'.AND. SUBSTR(ITEM,6,1) <> ' '
IF .NOT. EOF()
SKIP
ENDIF
ENDDO
X=X+1
ENDDO
ENDIF
IF NUM<F
GOTO NUM
ELSE
NUM=1
ENDIF
IF P=1
IF .NOT. EOF()
SET COLOR TO R/B
@ 8,3 SAY CODE
SET COLOR TO R/W
ENDIF
N=0
ELSE
N=0
SET COLOR TO R/B
DO WHILE SUBSTR(ITEM,6,1) <> '1'.AND. SUBSTR(ITEM,6,1) <> ' '
IF .NOT. EOF()
SKIP
NUM=NUM+1
ENDIF
ENDDO
@ P+7,3 SAY CODE
IF RECNO(<>) <> 1
NUM=NUM-1
SKIP(-1)
ENDIF
DO WHILE SUBSTR(ITEM,6,1) <> '1'.AND. SUBSTR(ITEM,6,1) <> ' '
IF RECNO(<>) <> 1
SKIP(-1)
ENDIF
ENDDO

```

```

NUM=NUM+1
SET COLOR TO R/M
@ P+6,3 SAY CODE
ENDIF
IF P=20
IF .NOT. EOF()
IF NUM<>1
SET COLOR TO R/M
GOTO NUM-1
IF RECNO() <> 1
ENDIF
DO WHILE SUBSTR(ITEM,6,1) <> '1' .AND. SUBSTR(ITEM,6,1) <> ' '
IF .NOT. EOF()
SKIP
ENDIF
ENDDO
@ 8,3 SAY CODE
GOTO NUM
ENDIF
ENDIF
NUM=NUM+1
IF NUM=F+1
RETURN
ENDIF
P=P+1
IF P=10
P=1
ENDIF
ENDIF
RETURN
* END LIST

```

```

      * LISTUP.PRG
* MOVE HIGHLIGHTED AREA BACK TO FORMER INFO.
DO BOX
IF NUM<F+1 .AND. NUM>1
GOTO NUM-1
SET COLOR TO R/M
IF P<2
P=10
SET COLOR TO R/B
@ 6,2 SAY ' PARAMETERS '
SET COLOR TO R/M
@ 16,3 SAY CODE
ENDIF
@ P-6,3 SAY CODE
SKIP(-1)
DO WHILE SUBSTR(ITEM,6,1) <> '1' .AND. SUBSTR(ITEM,6,1) <> ' '
NUM=NUM-1
IF RECHO()>1
SKIP(-1)
ENDIF
ENDDO
SET COLOR TO R/B
IF P<2
@ P-5,3 SAY CODE
ENDIF
NUM=NUM-1
P=P-1
ENDIF
IF P=1
P=10
SET COLOR TO R/M
SET COLOR TO R/B
X=1
IF NUM>9
GOTO NUM-9
ELSE
NUM=1
P=1
SET COLOR TO R/M
DO LIST
ENDIF
DO WHILE X<10
SET COLOR TO R/M
IF X=9 THEN
SET COLOR TO R/B
ENDIF
@ X-7,3 SAY CODE
IF .NOT. EOF()
SKIP
ENDIF
DO WHILE SUBSTR(ITEM,6,1) <> '1' .AND. SUBSTR(ITEM,6,1) <> ' '
IF .NOT. EOF()
NUM=NUM+1
SKIP
ENDIF
ENDDO
X=X+1
ENDDO
ENDIF
IF X=10 .AND. (SUBSTR(ITEM,6,1)='1' .OR. SUBSTR(ITEM,6,1)=' ')

```



```

X=0
NUM=NUM-1
ENDIF
SET COLOR TO R/N
RETURN
* END LISTUP

```

---

```

* SEARCH.PRG
* LOOK FOR MATCH OF INPUT
Z=1
IF NUM<F+1
GOTO NUM
ENDIF
SET COLOR TO G/R
@ 13,30 SAY ' SEARCHING FOR '
@ 13,46 SAY CHR(1)
SET COLOR TO R/N
DO WHILE CHR(1) <> SUBSTR(CODE,1,1)
NUM=NUM+1
IF .NOT. EOF()
SKIP
ELSE
* ERROR TRAP
SET COLOR TO G/R
@ 13,30 SAY ' NONE OF THIS TYPE : '
SET COLOR TO G/R
@ 13,52 SAY CHR(1)
DO WHILE Z<150
* PAUSE FOR TIME
Z=Z+1
ENDDO
RETURN
ENDIF
ENDDO
IF NUM<F+1
GOTO NUM
C=NUM+1
P=1
DO LIST
ELSE
* ERROR TRAP
SET COLOR TO G/R
@ 13,30 SAY 'ENTER ANOTHER COMMAND'
SET COLOR TO R/N
ENDIF
RETURN
* END SEARCH

```

```

      * LOOK.PRG
XX='Y'
Z=1
DO WHILE XX='Y'
  SKIP 1
  CODE2=CODE
  SKIP (-1)
  IF CODE=CODE2
    Z=Z+1
  ELSE
    XX='N'
  ENDIF
  SKIP 1
ENDDO
SKIP(-1)
DO WHILE SUBSTR(ITEM,6,1)<>'1' .AND. SUBSTR(ITEM,6,1)<>' '
  SKIP(-1)
ENDDO
SET COLOR TO G/M
ZZ=10
QQ=30
L=2
@ 6,26 SAY CODE
@ 6,47 SAY FAINTF
@ 7,47 SAY PORTCONN
@ 9,24 SAY SC_CHAN
@ 9,42 SAY SC_CARD
@ 9,64 SAY PS_EXCIT
@ 13,22 SAY SC_IP
@ 14,23 SAY RD_OP
SET COLOR TO R/N
@ 6,16 SAY 'PARAMETER'
@ 6,37 SAY 'INTERFACE'
SET COLOR TO G/B
@ 8,18 SAY 'CHANNEL INFORMATION'
SET COLOR TO R/N
@ 9,16 SAY 'CHANNEL'
@ 9,30 SAY 'CONDITIONER'
@ 9,53 SAY 'EXCITATION'
SET COLOR TO G/B
@ 12,18 SAY 'PATCHING'
SET COLOR TO R/N
@ 13,16 SAY 'INPUT'
@ 10,52 SAY 'FILTER'
@ 10,63 SAY 'GAIN'
@ 14,16 SAY 'OUTPUT'
SET COLOR TO G/B
@ 18,1 SAY 'AUX RECORDERS INPUT PATCHES'
@ 17,19 SAY 'RECORDERS'
SET COLOR TO R/N
N=1
SKIP
DO WHILE SUBSTR(ITEM,6,1)<>'1' .AND. SUBSTR(ITEM,6,1)<>' '
  SKIP
ENDDO
DO WHILE Z>0
  SKIP(-2)
  SET COLOR TO G/M
  @ 17,00 SAY RD_IDEN
  @ 18,00 SAY RD_IP

```

```

@ 10*N,52 SAY RD_FILTER
@ 10*N,63 SAY RD_FS
SET COLOR TO R/N
@ 16,00 SAY ITEM
@ 10*N,44 SAY ITEM
ZZ=ZZ-7
QQ=QQ-7
N=N+1
SET COLOR TO R/N
* @ 17,00 SAY ITEM
SKIP 2
QQ=QQ+20
ZZ=ZZ+26
Z=Z-1
ENDDO
XX='Y'
SKIP(-L)
RETURN
* END LOOK

```

```

* PAGE.PRG
* MOVE TO NEXT PAGE
NUM=NUM+7
IF NUM>=F
  NUM=1
ENDIF
GOTO NUM
P=1
DO LIST
RETURN
* END PAGE* PGUP.PRG
* MOVE TO LAST PAGE OF INFO.
NUM=NUM-9
IF NUM<1
  NUM=F-8
ENDIF
IF NUM>F
  NUM=1
ENDIF
GOTO NUM
P=1
DO LIST
RETURN
* END PGUP* PGUP.PRG
* MOVE TO LAST PAGE OF INFO.
NUM=NUM-9
IF NUM<1
  NUM=F-8
ENDIF
IF NUM>F
  NUM=1
ENDIF
GOTO NUM
P=1
DO LIST
RETURN
* END PGUP

```

```

* CHANGE.PRG
ANS='N'
DIF=SPACE(5)
SET COLOR TO G/N,B/G
@ 0,41 CLEAR TO 0,79
@ 20,0 CLEAR TO 23,79
@ 21,7 SAY 'COMMENT : ' GET COMMENT1
@ 0,41 CLEAR TO 0,79
@ 22,17 SAY '' GET COMMENT2
READ
DO LIST
RETURN
* END CHANGE

```

TITLE: POWER COORDINATION STUDY

Heather B. Hopwood

August 12, 1991

Mentors:

Bill Mayberry

Tony Acklen

Elizabeth Broyles

Arnold Air Force Base

Tullahoma, TN

The Power Coordination Study updates information on the protective function that a relay performs. Specifically the study provides information concerning the effectiveness of the protective devices. If a fault is found in the protection, modifications are recommended. A database filled with information about the relays, joined with Electrical Distribution System Analysis program (EDSA) will be able to display which relay at what facility is working ineffectively.

I show much gratitude to Bill Mayberry who selected me as his apprentice, and Tony Acklen and Elizabeth Broyles, who accepted the responsibility of guiding me through the study. A special thanks to Dale Coulson for assisting me with AutoCAD which I used most frequently during the course of the study. GB13 section should be recognized for their helpful guidance, and their encouraging attitudes.

The Power Coordination Study will accumulate new, and pre-recorded data. The old information is gathered from existing books, drawings, and files. The new data is taken from relays, and their readings. After all the information is accumulated and entered into the database, it must be printed out before entering the information into EDSA it will now be possible for EDSA to display a problem with one of the relays in the one-line diagram.

No research has been done concerning this subject in thirty years. The information needs to be updated to see if the relays are working efficiently. If the relays are not working correctly an overload could occur, and nothing would be able to protect the equipment from excessive damage or fire. When this study is completed a new study will begin. The new study, called Relay Modernization, will determine how to correct the faulty device. After all of the problems have been identified, recommendations will be made on how to correct the problem. This study will explain how to fix improper working relays.

A relay is an electrical device that provides the logic to tell a circuit breaker to trip. A circuit breaker is used to isolate the source of power from the load. A breaker does this to protect the load from various disturbances such as overloads, short circuits, and low voltage supply. Relays protect lines, transformers, and loads. Examples of loads are motors and unit substations. Loads are the devices that use the electrical energy. For example motors use the

electrical energy and turn it into mechanical energy. The mechanical energy is the power that generates the air that the wind tunnel uses.

Many relays are needed to protect a transformer. A transformer is an electrical device that transforms voltage from one level to another level. For example: the primary power that is distributed at AEDC is 161,000 volts. The transformer reduces the voltage to 13,800 volts. The relays protect the transformer from overloads, short circuits, and low voltage supplies. If these disturbances ever occurred, excessive damage would most likely occur if the relays are not functioning properly. Transformers reduce the voltage to a level suitable for use by the loads which is safe and efficient.

To start gathering data on a building, I would obtain drawings of the electrical layout of the building. Next I would read the drawings and fill out information concerning the relays. For example: location, name, device number, manufacture type, time characteristics, various settings of the relay, and potential and current transformer ratios. Then I would go out in the field to the location of the relays and write down the the relay type, style number, and instruction number. I would also check the information already written down. Afterwards I would enter all the information into the database which my co-worker and I designed for this study.

Many databases have been made for the study. All the information gathered from the relays had to be put into the database. Many of my



hours have been spent at the computer entering the information from drawings and old files. There is also a database made for all the motors on the base above one hundred horsepower. If a person is checking data, the database will be able to show the person: location, horsepower, RPM, base KVA, full load amps, power factor, positive sequence sub-transient reactance, X/R ratio, and voltage. Changes can be made to the database. For example: if a motor has been replaced or a new motor installed a new file can be created, and the old file destroyed.

I have also drawn one-line diagrams using Auto-CAD (Computer Aided Design). AutoCAD enables me to draw an electrical outline of a facility. EDSA has also been linked with AutoCAD. The EDSA menu has electrical symbols, and will label items according to the data entered. After everything is entered EDSA will evaluate the protection that relays provide.

Diagram I is an example of a one-line diagram. The diagram has been simplified to fit on a single sheet of paper. The one-line diagram still obtains the basic appearance of the original diagram. This one-line diagram is the electrical layout of Aero-Propulsion Systems Test Facility (ASTF). ASTF was the area my main priority was placed at. Transformer A-1 Primary reduces the voltage TVA sends at 161 KV to 13.8 KV. Then the transformer sends the voltage to different loads. If anything happens to the motor or transformer, circuit breaker 42-14B will shut down the load. This will only take place if

the relays sense the disturbance in time.

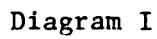
Diagram II is the output that EDSA has given me concerning ASTF. The small highlighted line represents motor M14 full load current. If motor M14 gets overloaded the current will rise above the present value, and the relays should operate to trip open circuit breaker 42-14b. On the graph it shows that the relays are working correctly. Relay 50/51 will activate as soon as it senses a disturbance. Relay 151T will replace 50/51 if it does not work, and the same process will happen with relay 51.

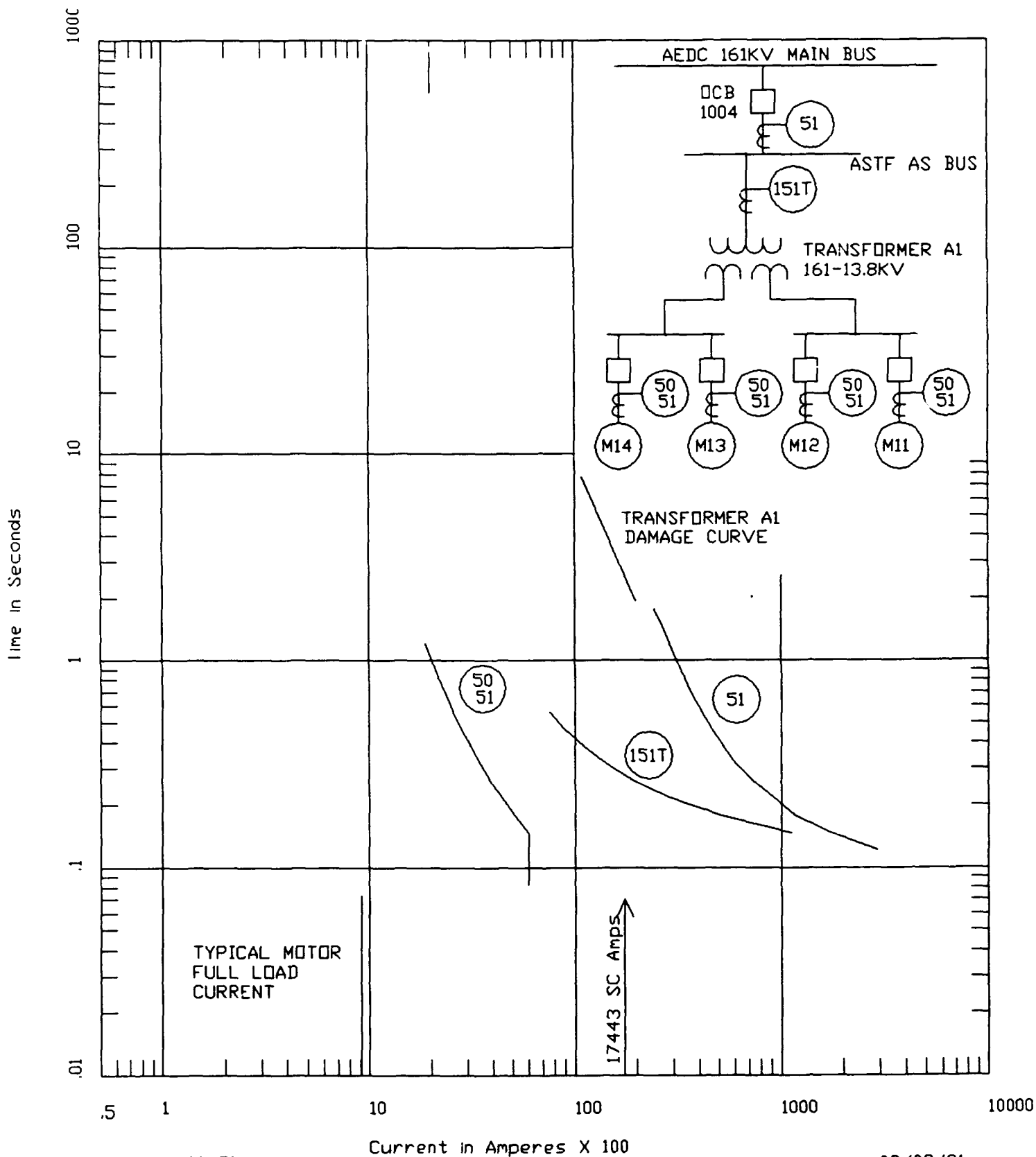
This graph shows that the relays connected with motor M14 are working correctly. If the relays were not in working order, and motor M14 was damaged it would be very expensive to repair. Motor M14 is rated 27,500 horsepower, and would cost at least a million dollars to replace it. This study will save dollars, if the relays are replaced before damage occurs to the motors.

There are many things I worked on this summer concerning the Power Coordination Study. A lot of effort went into the beginning stages of this study. I had to learn how to draw a simple line with AutoCAD before I could draw a one-line diagram. I had to learn what a relay was, and what it looked like before I could read one. There have been simple things I have done just to get started on the study. They took their time to show me what I needed to accomplish.

This summer I have learned many new things. I now know the advantages and disadvantages of being an engineer. This summer's experience has made me want to strive for the best in school, and except no less. I enjoyed being a part of this program, and hope to come back next summer to work again.

The Power Coordination study has successfully completed a section on the facility ASTF. According to EDSA, and all the gathered data, the relays at this section are working correctly. Information also involving other sections on the base have also been collected. EDSA will later process the other data and determine relay reliability.





13800 Volt Phase  
Time-Current Characteristic Curves  
ASTF AIR SUPPLY

08/02/91  
12:56:57

Daigram II

**DEVELOPMENT OF FINITE ELEMENT MODELING  
PROCEDURES**

Kevin Johnson

Kurt L. Nichol, Mentor

Arnold Engineering Development Center

9 August 1991

## DEVELOPMENT OF FINITE ELEMENT MODELING PROCEDURES

Kevin L. Johnson

### Abstract

The objectives of this research were to 1) develop and demonstrate finite element modeling procedures and 2) evaluate a recently acquired personal computer based finite element program called Algor. After learning the modeling procedures, the engine component was modeled. After defining the geometry and simulating component interfaces a stress analysis was performed. Results of this effort include evaluation of Algor program specific features and initiation of the analysis of the selected engine component. These results will eventually help to define finite element modeling procedures.

### Acknowledgements

The researcher would like to express his appreciation to the United States Air Force, Research and Development Laboratories, and Sverdrup Technology for providing this High School Apprenticeship Program. Additionally the researcher would like to thank Mr. Kurt L. Nichol for his guidance and direction which enabled the researcher to accomplish this analysis. The researcher would also like to thank Ms. Dollena S. Hawkins for her help on this analysis. Finally, thanks is extended to the rest of the supporting cast, Mr. William W. Muse and the Sverdrup Technology employees in office building 935 at Arnold Air

Force Base, Tennessee.

### General Description

A gear which is in the main fuel pump of the F110 Turbofan Engine was modeled using the Algor Finite Element Analysis System. Three different load cases were applied to the gear model. Each of the three loading conditions were analyzed separately. Results of these analyses will be used to support a substructure analysis which will improve the accuracy of the final results. Modeling of the substructure model is complete. The final outcome of this work will enable engineers to have a basic idea of how the Algor Finite Element Analysis system actually works and enhance the speed of future research efforts.

### Detailed Description

Finite element analysis is a numerical method used for analysis of complex structural engineering problems. The basic idea of the finite element method is to reduce a complex problem (geometry) into a large number of smaller, simpler problems. The building blocks used to represent the complex structure are called finite elements. Because of the simplification of the problem, the results represent only an approximate solution rather than an exact solution [ Refer to figure 1 for a visual image of finite elements]. The basic objective of this analysis was to develop and demonstrate modeling procedures using a recently acquired personal computer based finite element



analysis program called Algor. The component to be analyzed was a gear in the main fuel pump of the F110 Turbo-fan Engine. The Algor program was used to model the gear and generate the stress fields resulting from various loading conditions.

The first step toward accomplishing these objectives was to get acquainted with the Algor system. The basis of the program is the graphics module, called Superdraw. In Superdraw, the geometry of the model was defined. Like finite elements, the basic structures are easy to work with, so to accomplish learning Algor the researcher and mentor developed a simple model to "practice" using Algor. After getting the basic idea of Algor and knowing how to operate the program, the researcher then began the process of analyzing the gear part.

The graphics module of Algor was used to define the geometry of the gear. Geometric coordinates were generated by the researcher's mentor, Mr. Kurt Nichol. Next, the researcher had to simulate the component interfaces by the addition of finite element analysis features such as loads and boundary conditions. Because these are independent of the geometry, they were applied as various load cases. Since the loading mechanism was very complex, the researcher had to run a parametric study of the various load cases to understand the sensitivity of the resulting stresses to changes in the load distribution. Each load case represented a different load distribution. The load

case which produced the most severe stress condition was selected for use in subsequent analysis due to the fact that it represented the worst case condition. For the required load conditions, several iterations of boundary conditions were applied to capture the physical characteristics for each load condition. In other words, the results had to make sense from a physical standpoint. After defining the geometry and applying the features, it was necessary for the researcher to clean the model of unnecessary duplicate lines and vertices generated during the model construction phase. The cleaning of the model would allow the geometry to be transformed accurately into individual finite elements. This was a program specific step unique to the Algor system.

Another module of the Algor finite element software was used to create the actual finite elements. This module was called the decoder. Material properties, body forces, and load case multipliers are defined here as well. The output of the decoder module was the finished finite element model which could then be used for structural analysis. The decoder took about eighteen (18) minutes to prepare approximately 1250 elements. After decoding, the researcher found it a good idea to look at the model in another module of Algor called Superview, to verify that the model was created correctly. If there was a problem, the researcher could go back to Superdraw and correct the problem. If the geometry was okay, the analy-

sis could proceed.

The next step was to run the model through the static analysis processor. The static analysis processor formulated equations corresponding to the various degrees of freedom, which were described by the finite elements. The analysis processor then solved these equations to produce the stresses and displacements at each node point. The static analysis processor took about three (3) hours to run. The results of the analysis were then read into the Superview module for interpretation by the mentor and the researcher. Figure 2 provides a visual description of the modeling steps. Hardcopy output was available from the Superview file by running the Superplot module of Algor.

### Results

Figure 3 represents the simple model which the researcher used to get acquainted with the Algor System. The results which were obtained from this model were compared to classical results, which were calculated by the researcher's mentor. This process allowed the researcher to find several modeling hints, which were helpful in subsequent efforts. The results of the model compared very well with classical methods.

Figure 4 is a graphic representation of the engine component. The model is shown here with the addition of boundary conditions. The picture shown was obtained using the Superplot module of Algor.

Figure 5 shows the Axial Force distribution of the three load cases. The plot shows the normalized force on the Y-axis verses the component Z-axis. By running these three load cases it was found that load case 2 provided the most severe stress distribution. Results from this load case were then selected for further analysis on the substructure model.

The substructure is shown in Figure 6. The model will allow continuing refinement of the analysis. The substructure has more refined geometry and it has increased output. The boundary conditions of the substructure were taken from the output of the large model for load case 2. The substructure has a notch cut from the center.

### Conclusions

The Algor Finite Element Analysis Program gave results from analysis of the simplified structures which compared favorably to classical results. Furthermore, by analyzing the gear, initial demonstration of a finite element analysis procedure was accomplished. Additionally, the researcher concludes that conducting a parametric study, as was done with the various load distributions, can be a valuable tool in other analyses.

### Observations

The researcher was very thankful for the opportunity to run this finite element analysis. By running this analysis, the researcher was given insight into the field of

engineering. By running this analysis the researcher was able to learn to use the Algor Finite Element program to perform finite element analysis.

**FIGURE 1**

The Finite Element Method in Engineering

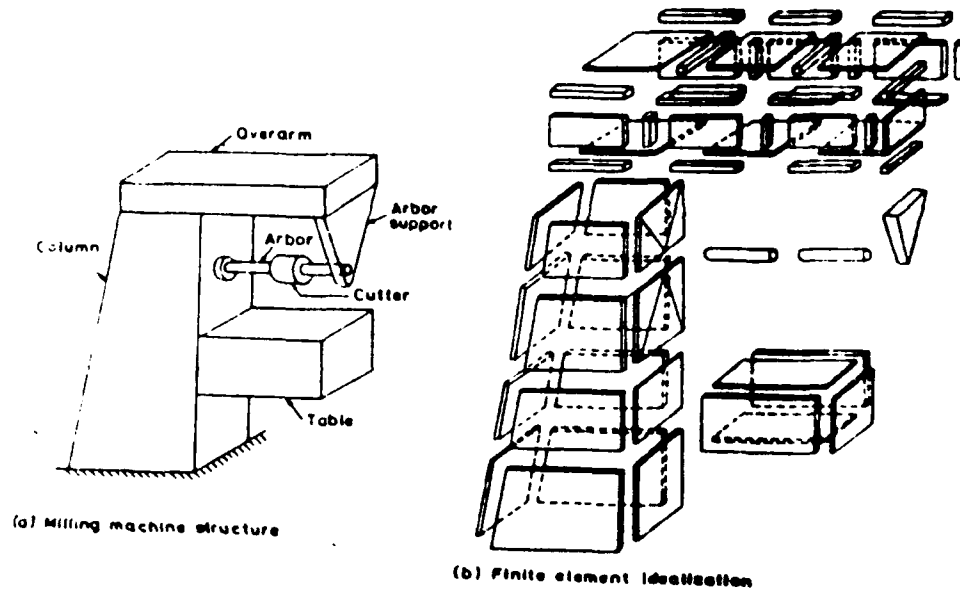
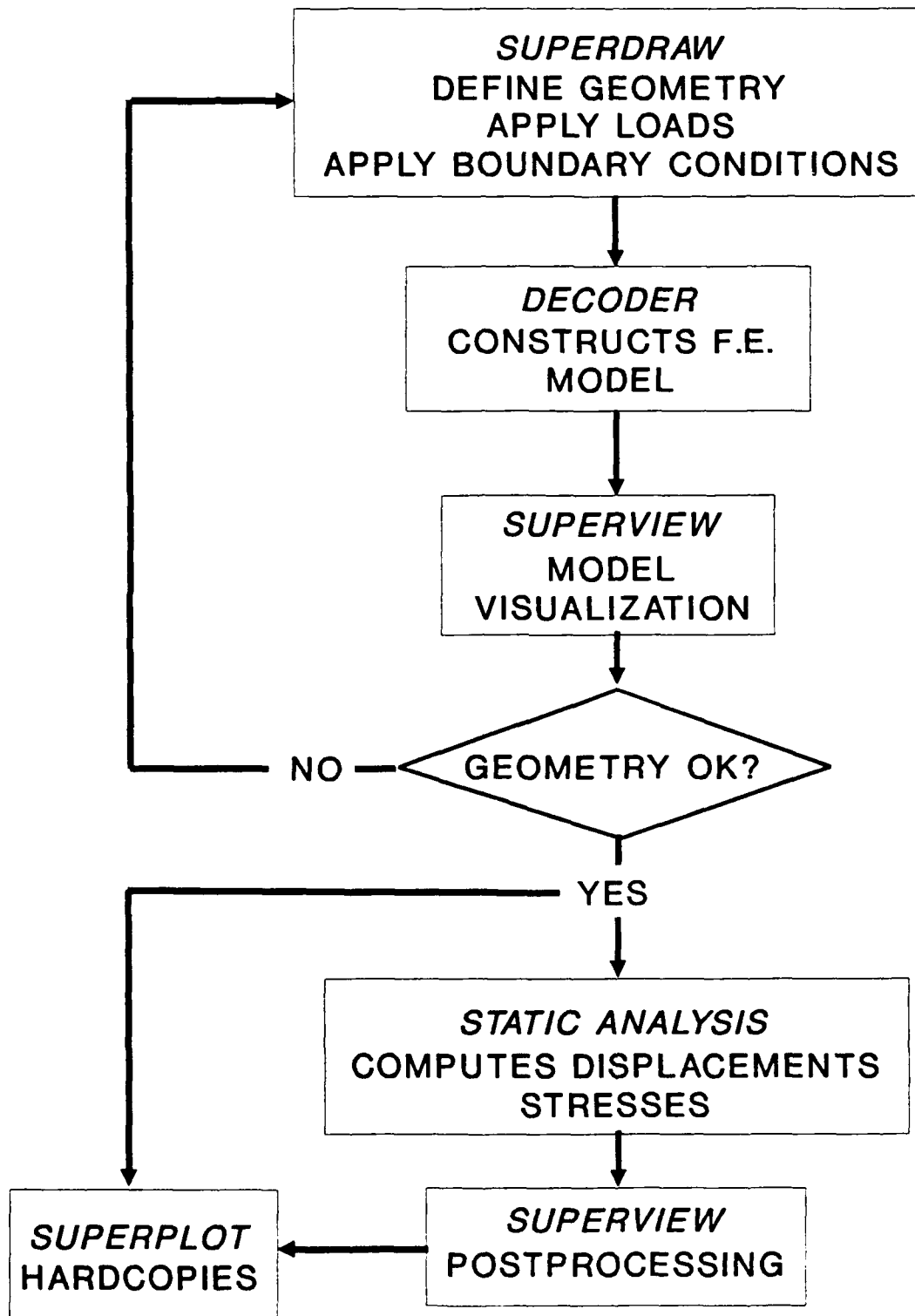


FIGURE 2



**FIGURE 3**

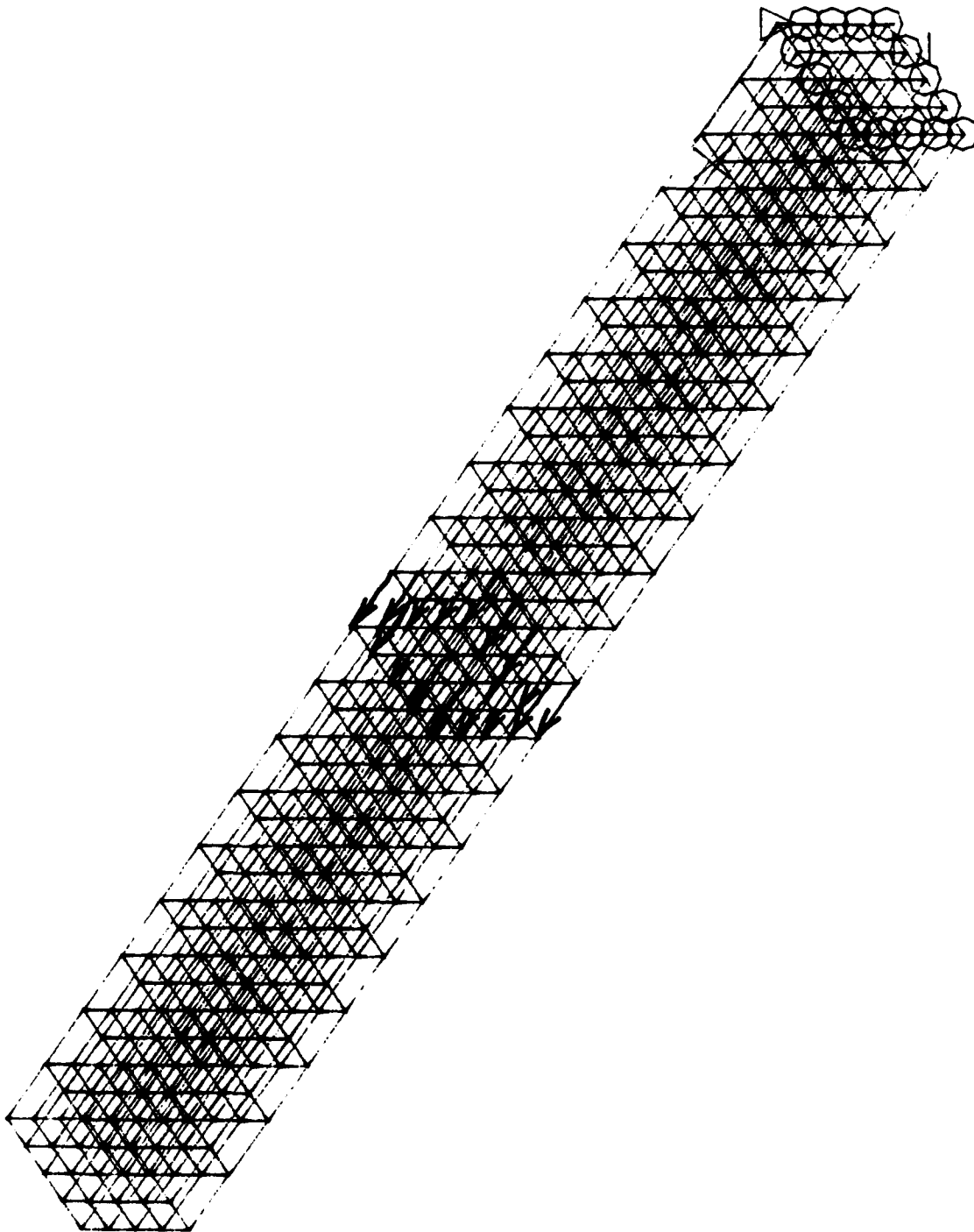




FIGURE 4

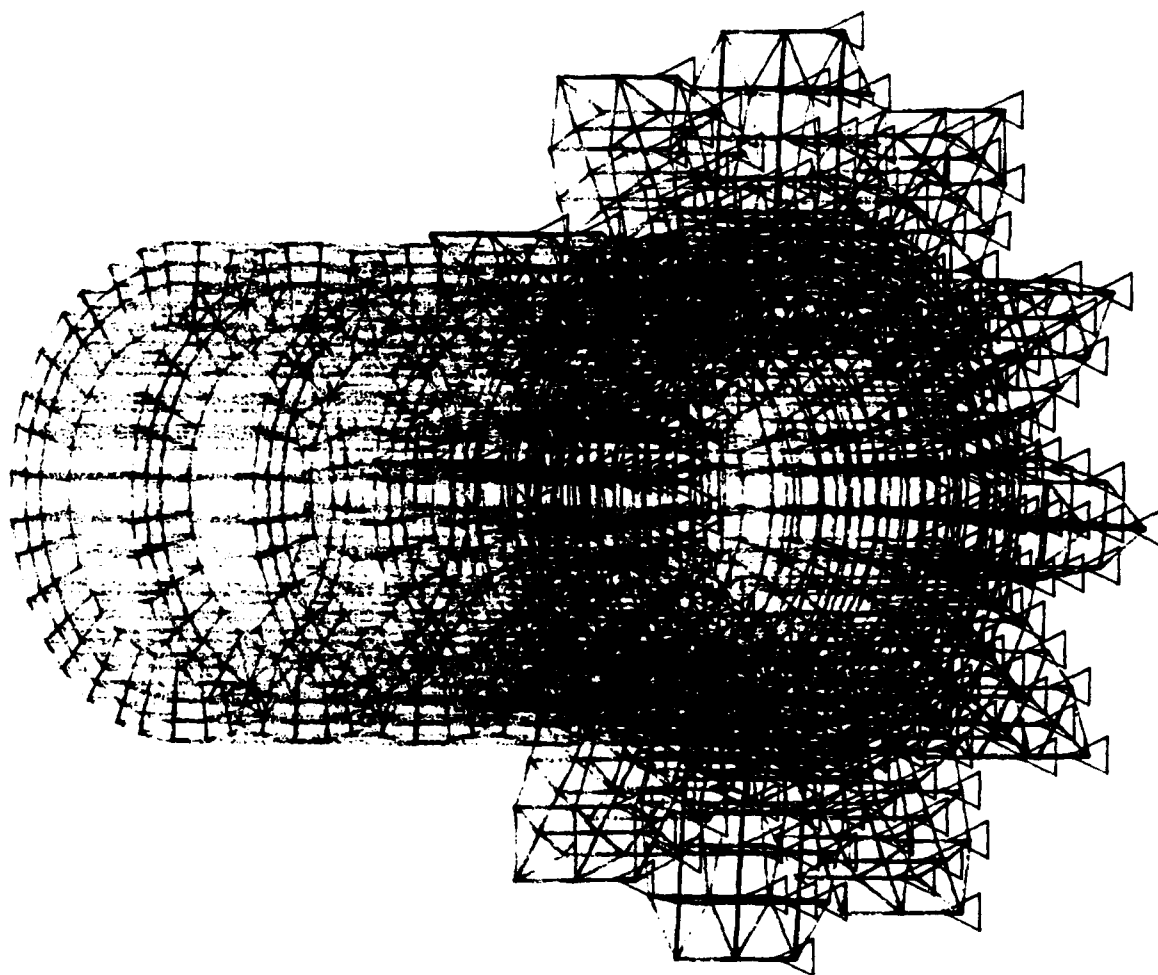
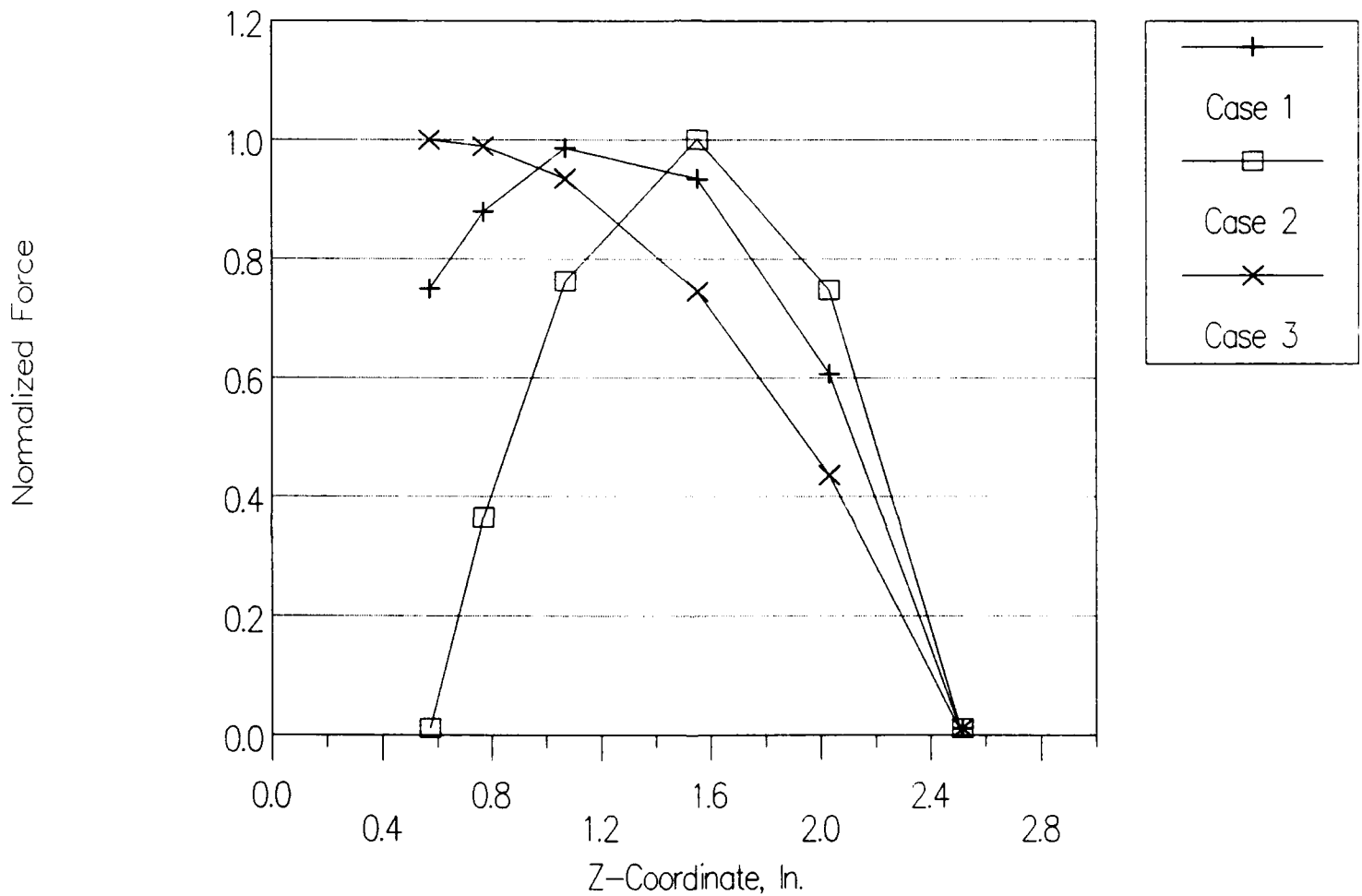


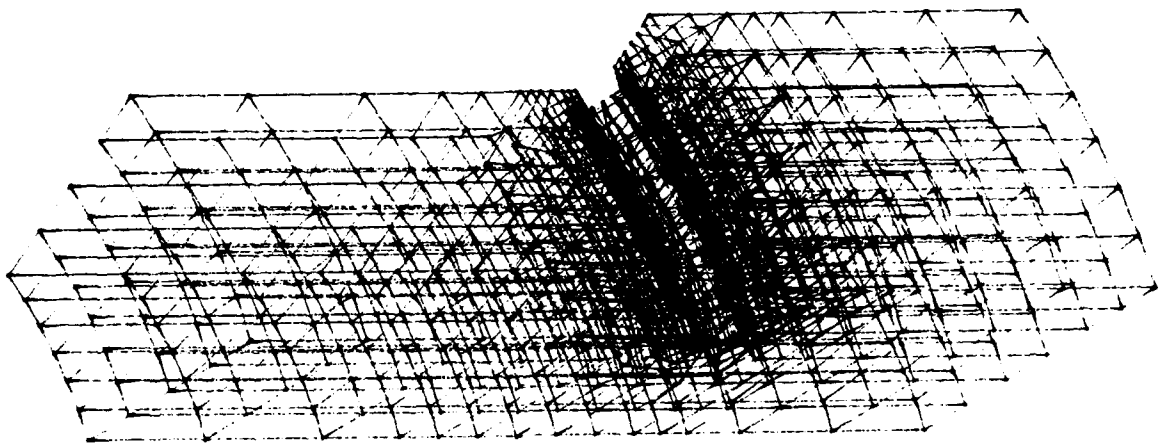
FIGURE 5

# Spur Gear Axial Force Distribution

All Three Load Cases



**FIGURE 6**



# **PLUME CHEMISTRY DIAGNOSTICS**

BY

Lisa G. Lewis

## ACKNOWLEDGMENTS

I extend a warm note of thanks toward Jim Drakes, Robert Hiers, Rita Walker, and Martha Simmons for their support and encouragement throughout my experience at AEDC. I appreciate each of their efforts for treating me as an equal, and not as an inexperienced high school student. I will always remember the kindness shown toward me as well as the knowledge I've gained in this experience.

## ABSTRACT

The testing was divided into two phases in which the writer participated. The first phase, computed tomography, was a diagnostic, nonintrusive technique which can be used to scan turbine/rocket plumes to measure the concentrations of species that determine the performance of the turbine/rocket. The second phase, ultraviolet mechanisms, was the actual test which combined with phase one. The author's involvement in the Ultraviolet Mechanism experiment was to determine whether or not a plasma could be generated in an atomic oxygen generator during a simulated combustion test.

## I. COMPUTED TOMOGRAPHY

### A. INTRODUCTION

The mathematical concept of tomography was discovered by Dr. J. Radon in 1917. The X-ray computed tomography (CT) systems were created in the late 1960's and the early 1970's because more powerful computers became available. (Ref. 1) Computerized assisted tomography, CAT, scanners were invented in the early 1970's by G. Hounsfield at EMI. Ltd., in England. The inventors of the CAT scanner received the Nobel Prize in 1979. Over the years, CAT scanners have spread over the world to help doctors diagnose various medical problems. Within the last five years, industries have started using CT systems for inspecting articles ranging from printed circuit boards to rocket motors. Computed tomography is not limited to X-ray radiation. The theory works for other forms of radiation such as visible, ultraviolet, and infrared light. These wave lengths are more useful for plume chemistry diagnostics of rocket motors and turbine engines. Although scientists have created other methods to test rocket exhausts, theoretically, CT will be the futuristic method for testing turbine engine exhausts.

The performance problems and defects of many engines are determined in the test cells of AEDC before the actual engines are installed in the aircraft. The Air Force has a need for this research because it is their responsibility to have safe and efficient aircraft. During the stages of development of the engine, the Technology Group of Sverdrup Technology, Inc. develops diagnostic techniques to be used in the test cell. The application of CT to plume chemistry diagnostics is in the early stages of development at AEDC.

## B. APPROACH

In the project on which the writer was involved, the technique of tomography based on visible light rather than X-ray radiation was being developed to scan exhausts from rockets, jets, etc., to detect elements and compounds found present during the firing of an engine. A flat flame burner (Fig. 1) was used to simulate an engine's exhaust. The burner burned hydrogen, which contained an abundant amount of sodium, and oxygen. The goal of this project was to see exactly how sodium was distributed throughout the flame of the flat flame burner. When the flame was scanned in this particular project, different pictures were taken by a CCD video camera at different angles of the flat flame burner. The information collected was then loaded into a computer program which reconstructed the information which told the absorption coefficient of the flame. The absorption coefficient gave a general indication of the amounts of sodium and its location. This information will be used to calculate the actual concentrations of the sodium at any point in the flame.

## C. DETAILED APPROACH

The writer arrived at AEDC at the opportune time because a new set of data had arrived. Before any of the actual tomographic analysis was performed, the writer was instructed on the operating system of MS-DOS, to learn the fundamentals of using these computer programs. The first step of the experiment was to digitize the videotaped flame in the Technology group's video lab using the Quantex image processor. The purpose of digitizing the videotape was to convert the intensity of the flame to number values. The Quantex captured images and stored them on floppy disks. The floppy disk

was then loaded on an IBM compatible PC to be run through a special program. This program selected a profile, centered, graphed, named, and created data files for each individual image at 18 different angles. Each data group was sliced at a different location to determine the amount of sodium at different heights within the flame. When the data were uploaded to the AMDAHL, AEDC's scientific mainframe computer, it reconstructed the images using the Donner algorithms written at the Donner Laboratory (Biology and Medicine Division) at the University of California (Ref.2). The Donner algorithms output a two dimensional grid of numbers. These numbers were then downloaded to the PC from the AMDAHL. The operator was able to display the images by assigning different colors to the numbers to see the distribution of sodium through the computer's color code. The images were displayed in colors because the human eye can discriminate colors better than shades of gray.

#### D. RESULTS

Figures 2,3,4, and 5 were reconstructed from eighteen different views around the flame. The images were slices at different heights in the flame. Although a rectangular image was expected, a horseshoe shaped image was reconstructed because the sodium was not distributed uniformly. Some of the reconstructed images show the effects of the tube that was inserted in the flame. The tube disturbed the flow of the sodium in the flame.

Other data were acquired without the tube to compare the two sets. The sodium seemed to be more concentrated on one side of the burner in comparison with the other two sides which consisted of significant amounts



of sodium. The center and one of the sides of the flame had no apparent sodium present. There was no evident reason for this unusual distribution of the sodium. When the height above the burner increased, the sodium decreased. At the highest height analyzed, the sodium level on the two sides decreased to a small amount. In conclusion, this technique seems promising to measure turbine exhausts constituents and their distribution.

## II. ULTRAVIOLET MECHANISM EXPERIMENT

### A. INTRODUCTION

The second phase of the plume chemistry diagnostics was the test denoted as the Ultraviolet Mechanism Experiment, conducted in the test cell R2H. One important aspect of this experiment was to use a radio frequency voltage source to dissociate molecular oxygen into atomic oxygen ( $O_2 \Rightarrow O + O$ ). The resulting atomic oxygen was then titrated into the exhaust of a small gaseous combustor (Fig. 6) where it reacted with the plume species and produced visible and ultraviolet radiation as shown in figure 7. The following propellant combinations were tested: fuel 86%  $C_2H_4$  + 14%  $NH_3$ , oxidizer  $O_2$ ; fuel 86%  $H_2$  + 14%  $NH_3$ , oxidizer  $O_2$ ; fuel  $H_2$ , oxidizer 80%  $O_2$  + 20%  $N_2O$ . The different titration species that were used to interact with the propellant combinations were NO, O,  $O_2$ , CO, and  $N_2$ . When this interaction occurred, a chemiluminescent reaction was formed which produced heat. The advantage of the RF (radio frequency) system over many atomic oxygen generator systems was its simplicity and utility. The R2H test cell was used because of its efficiency in performing high altitude simulations on combustion.

## B. APPROACH

The student's responsibility in this experiment was to aid in the development of the plasma generation system. The first days of the student's apprenticeship involved working with a team to develop an atomic oxygen generator which would produce a plasma to detect whether or not certain materials would withstand high temperatures. The structure of the generator involved the placing of voltage across an impedance load, which included a coil which generated electric and magnetic fields in its center. The purpose of using a plasma source was to separate the  $O_2$  into individualized oxygen atoms which caused the reaction in the plume. The oxygen molecules were separated using radio frequency energy. The setting for the RF generator for the breaking of the bonds was 13.6 MHz. After the atomic generator was prepared, the team had to determine by process of elimination the type of tubing which would withstand the high temperature of the plasma. The first tube used, which was made of glass, melted under the intense heat from the plasma. The second tube used, which was made of quartz, didn't melt under the heat. Cooling air was added to this experiment to run through the coils on the atomic oxygen generator to decrease the temperature of the coils as shown in figure 8. The test cell instrumentation consisted of the following: Datametric transducers which were used to measure the pressure in the chambers, UV Spex Spectrometer which recorded emission spectra in the 209 to 241 nanometer range, VUV McPherson Spectrometer which recorded emission spectra in the 150 to 270 nm range, a high resolution camera which photographed the small details of the changes in the plume during firing, and a video recorder which recorded the events in the test cell. See figure 9.

### C. DETAILED APPROACH

The student was taught the fundamentals of the experiment before any actual testing was done. Rules on safety were also discussed before the student was allowed to enter the test cell area. The first few weeks of the student's apprenticeship dealt with helping the other workers on the team develop a more efficient atomic oxygen generator. When new ideas were suggested for the tubing of the generator, the student assisted. When the time came to enter the test cell area, the student and other team members worked on creating a more efficient generator which would decrease the chances of overheating the coils and tube which were located near the plasma. Much of the work that was done in the lab had to be changed when the generator was taken to the test cell, because the tubing didn't fit in the space provided, and the tubing didn't fit many of the valves. After these problems were corrected, the testing began. During testing, the student recorded data and video recorded actual test runs from the control room during the tests. The student also operated the computer system for the RF generator which powered the generator on or off and increased or decreased the power supplied to the system. The power setting was usually set at 400 watts because the plasma appeared at this power setting. If the system was not fine tuned, the reflected power (power which is reflected from the impedance load) usually appeared above 76 watts. If the reflected power was over 76 watts, the system had to be tuned until the reflected power appeared below 76 watts. An average setting for a typical run was a power setting of 400 watts, a reflected power setting of 29 watts, and a pressure setting of 8.2 torr.

## D. RESULTS

The atomic oxygen generator performed exceptionally well in this experiment. The appearance of a plasma in the atomic oxygen generator during testing was a great asset to the performance of the Ultraviolet Mechanism experiment.

Some of the runs lasted for 20 seconds while others lasted for 30 seconds. The main goal of my involvement was to produce a plasma in the generator during combustion. This goal was successfully achieved. The nature of the reaction of atomic oxygen with various plume species is now undergoing further investigation.

## III. OBSERVATIONS

An important lesson in this project was the use of a new computer operating system. Through learning MS-DOS, the writer received experience with both of these projects. The writer also received hands-on-experience with the Quantex image processor. A systematic analysis of the data was also learned during these projects. Word processing was learned during the last phases. Many lessons were also learned in the Ultraviolet Mechanism experiment. Not only was the student taught the fundamentals of the project, the student also participated in many of the decisions of the project. The student worked with the computer system in this experiment as well as the construction of the atomic oxygen generator. Other work included recording data, video recording test runs, and suggesting innovative ideas to the project team. Two different means of measuring plume species were the titration method and

the CT method. Both methods have strong and weak points. The strong point to titration was that it was a conceptually straightforward concept. The strong points to CT were that it had well developed theories and algorithms. The weak point of the titration method was that the calibrations were frequently inaccurate. The weak points of CT were that the instrumentation wasn't fully developed and the complexity of the system. Both methods are means to the common end of determining species' concentrations in plumes.

#### IV. REFERENCES

1. Dennis, Michael, "Industrial Computed Tomography", Metals Handbook, Vol. 17, 9th Ed., ASM International, Materials Park, Ohio, 1989, PP. 358-386.
2. Huesman, R.H., et al., "Donner Algorithms for Reconstruction Tomography," Users Manual, pub 214, Lawrence Berkeley Laboratory, University of California, 1977.

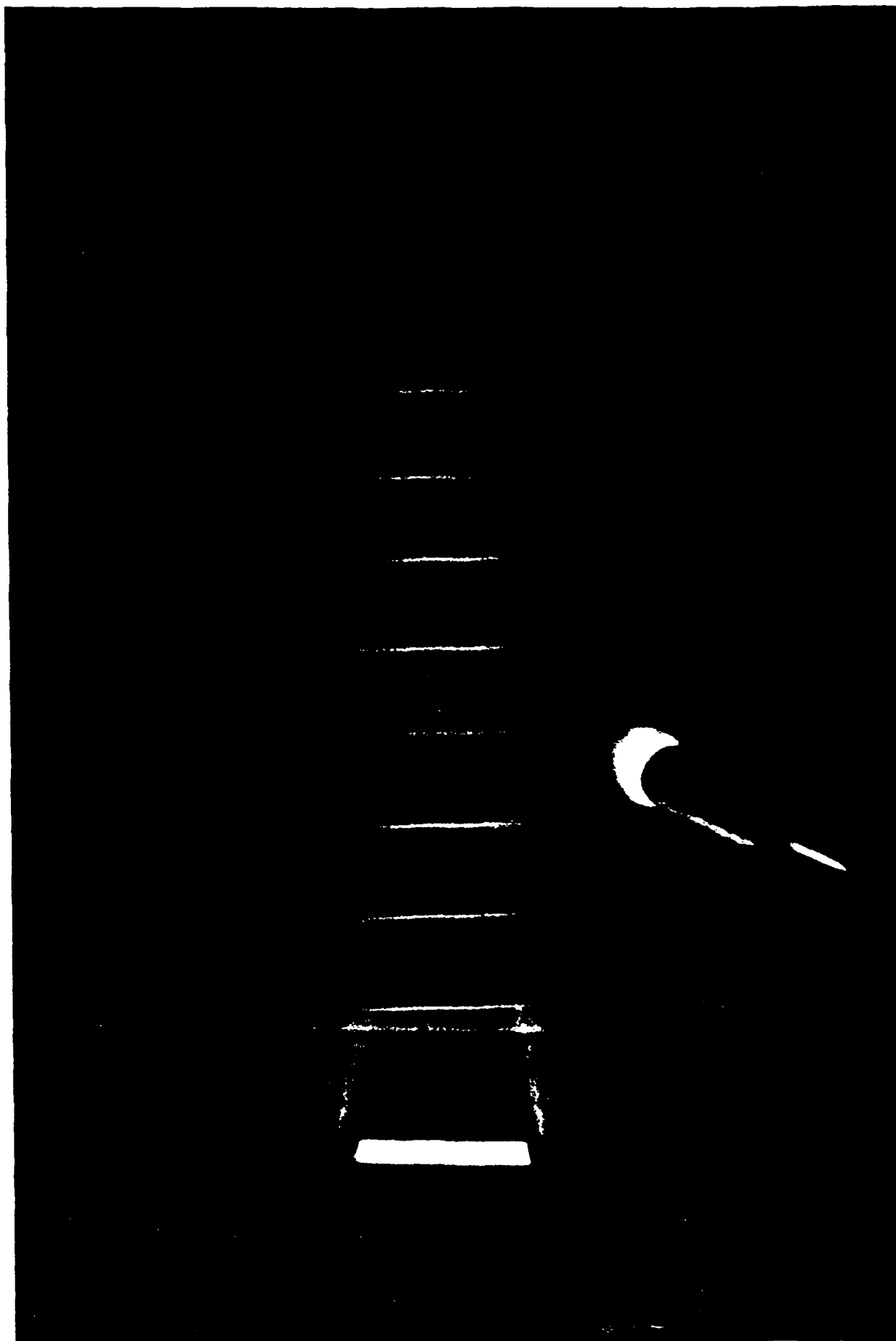


Figure 1. Flat flame burner.

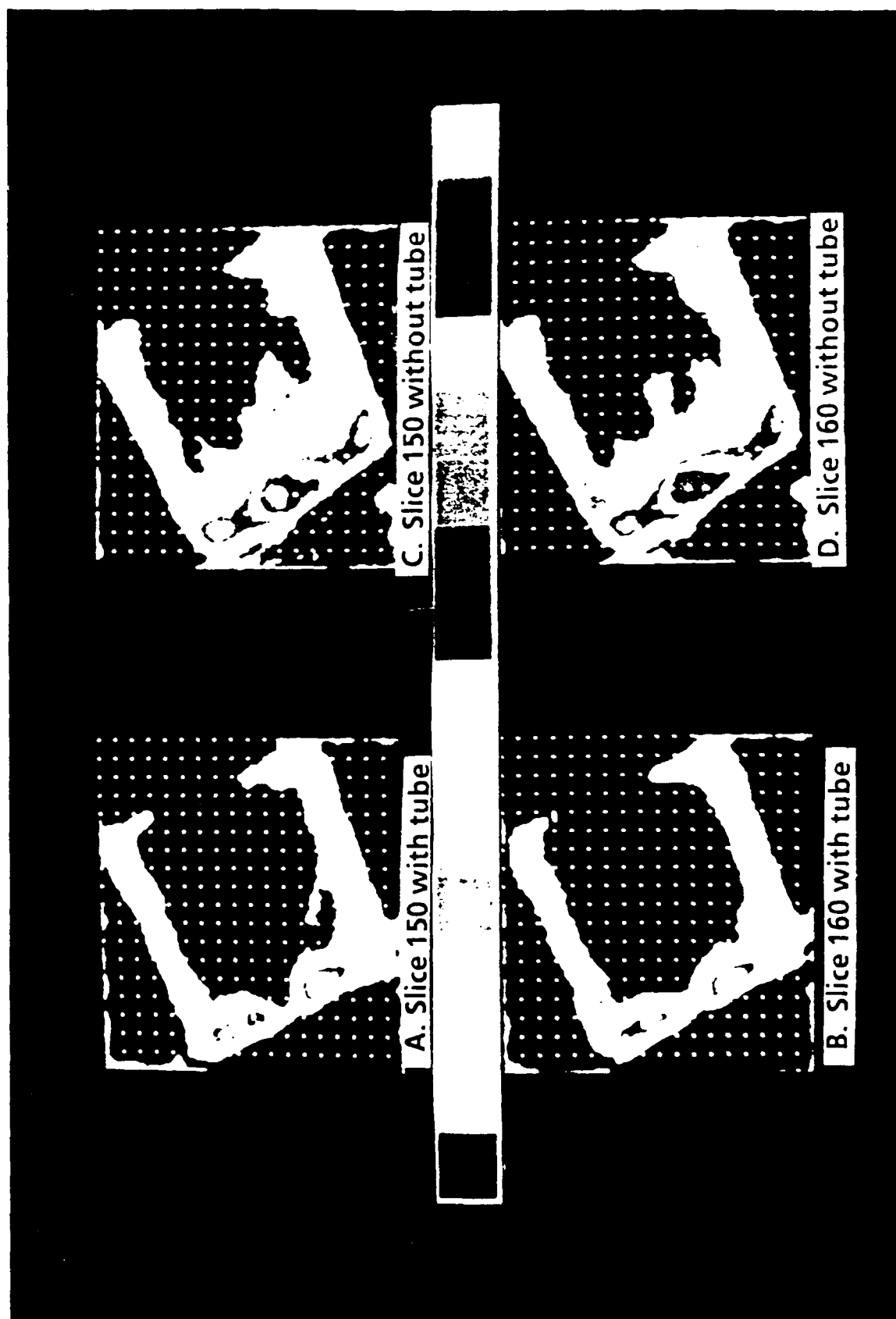


Figure 2. CT Reconstruction of flat flame burner.

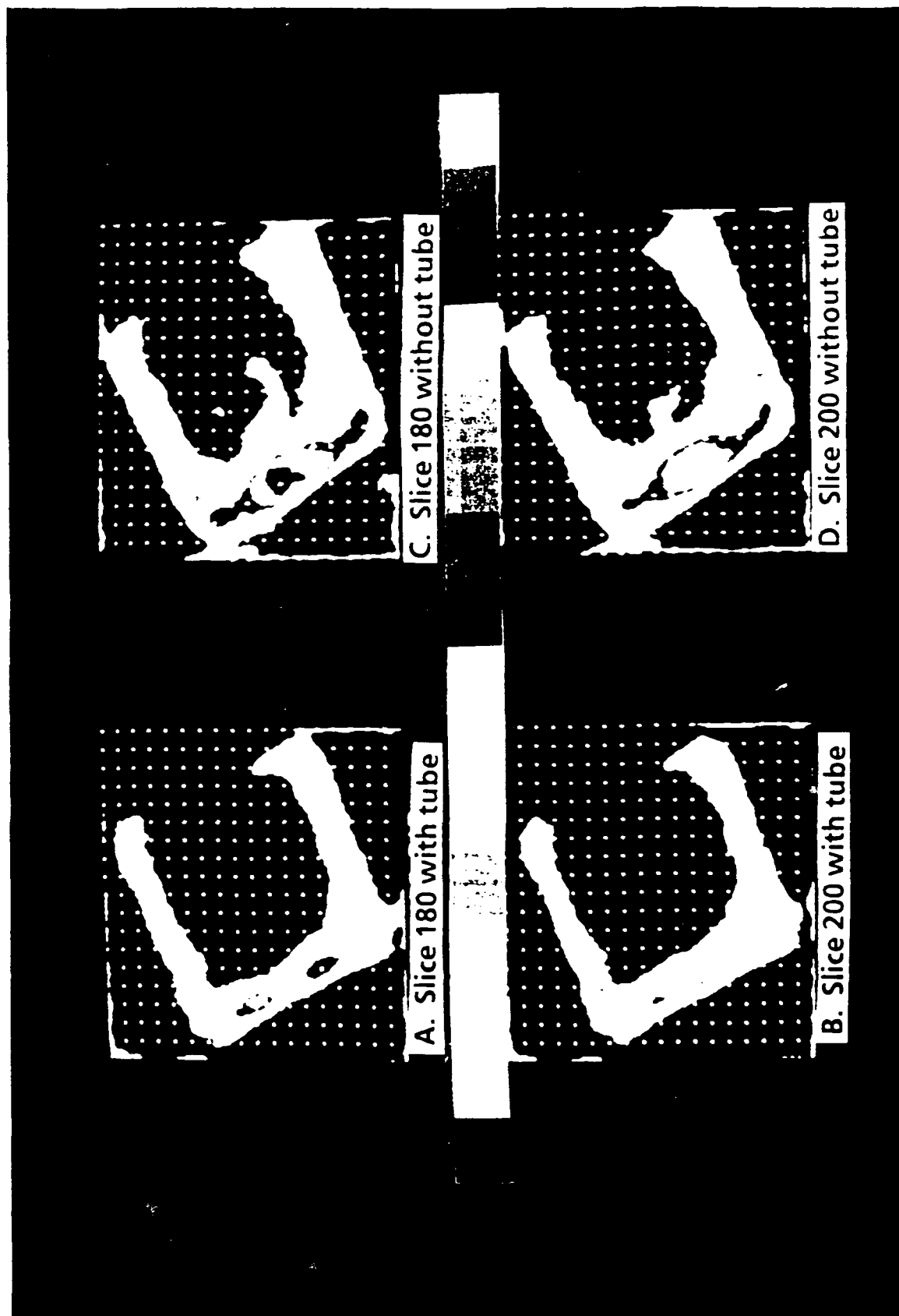


Figure 3. CT Reconstruction of flat flame burner.



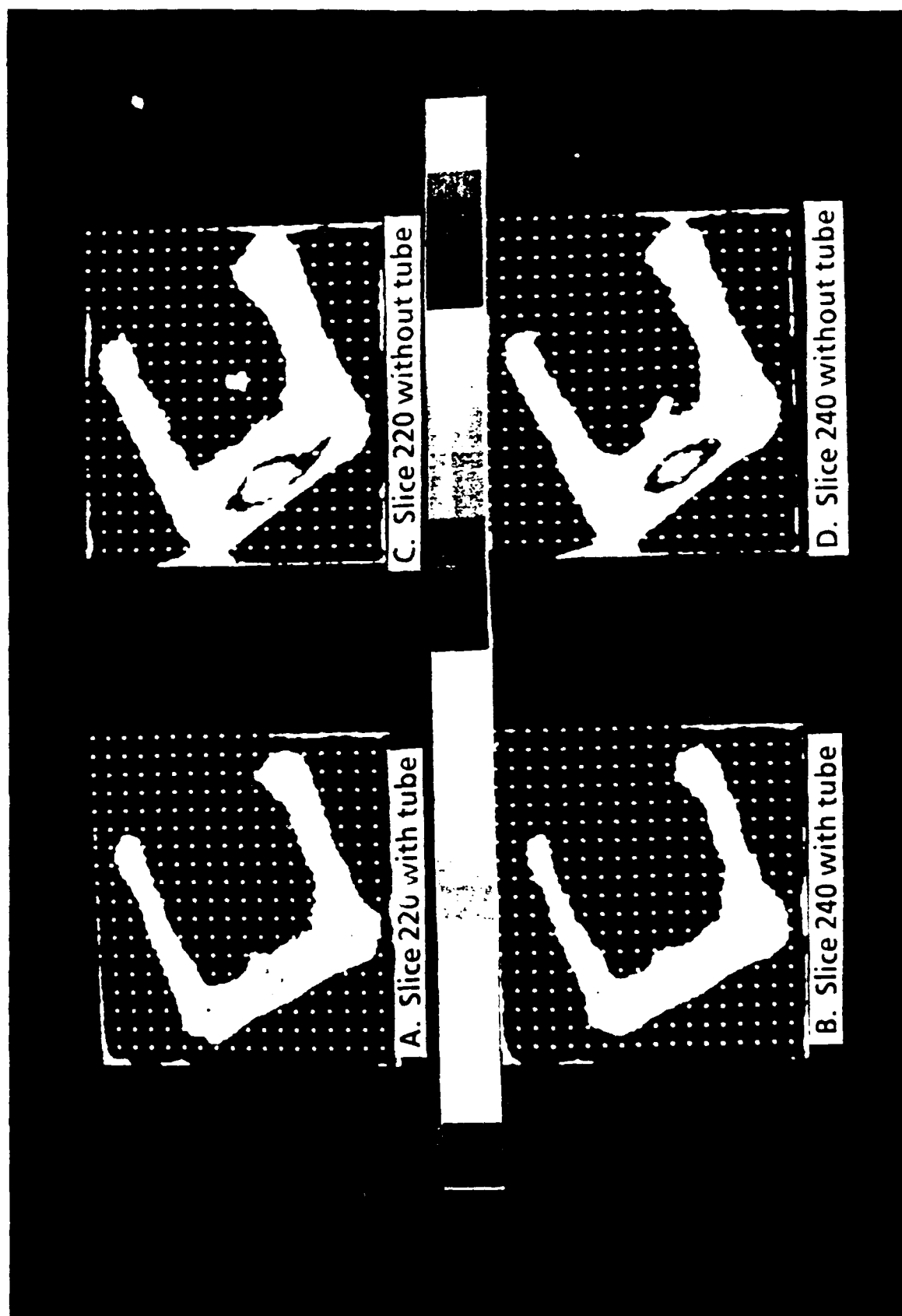


Figure 4. CT Reconstruction of flat flame burner.

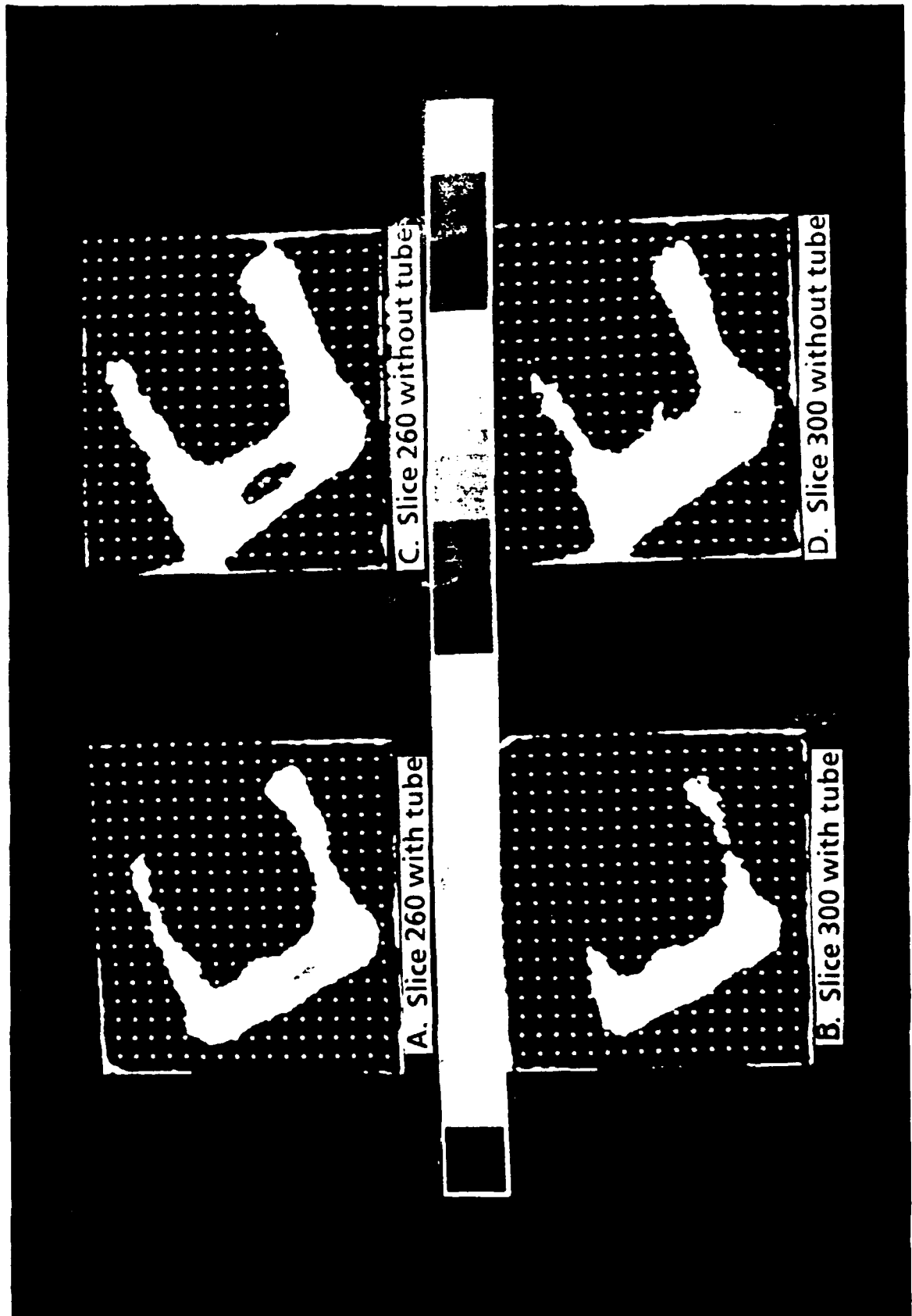


Figure 5. CT Reconstruction of flat flame burner.

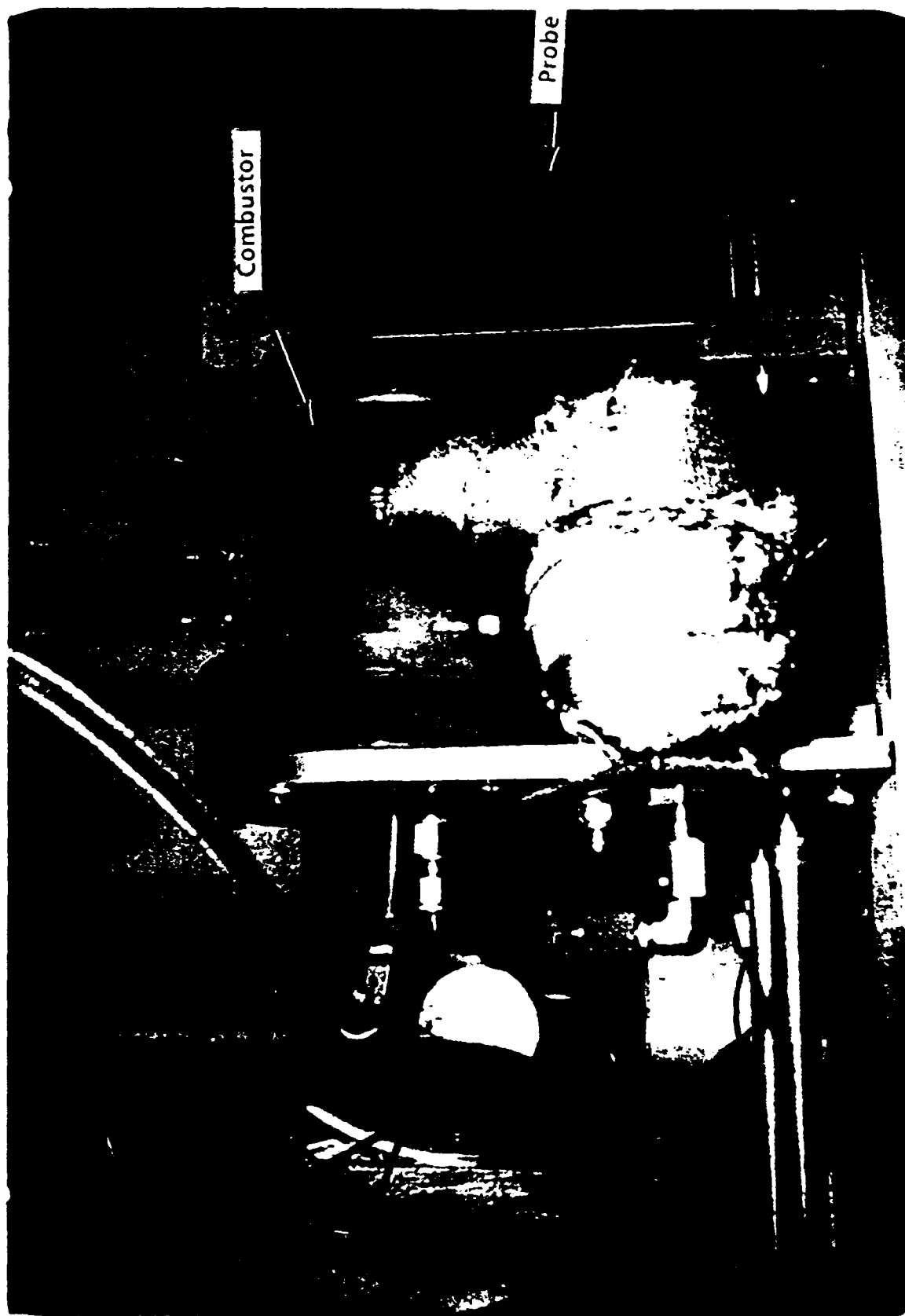


Figure 6. Combustor and probe.

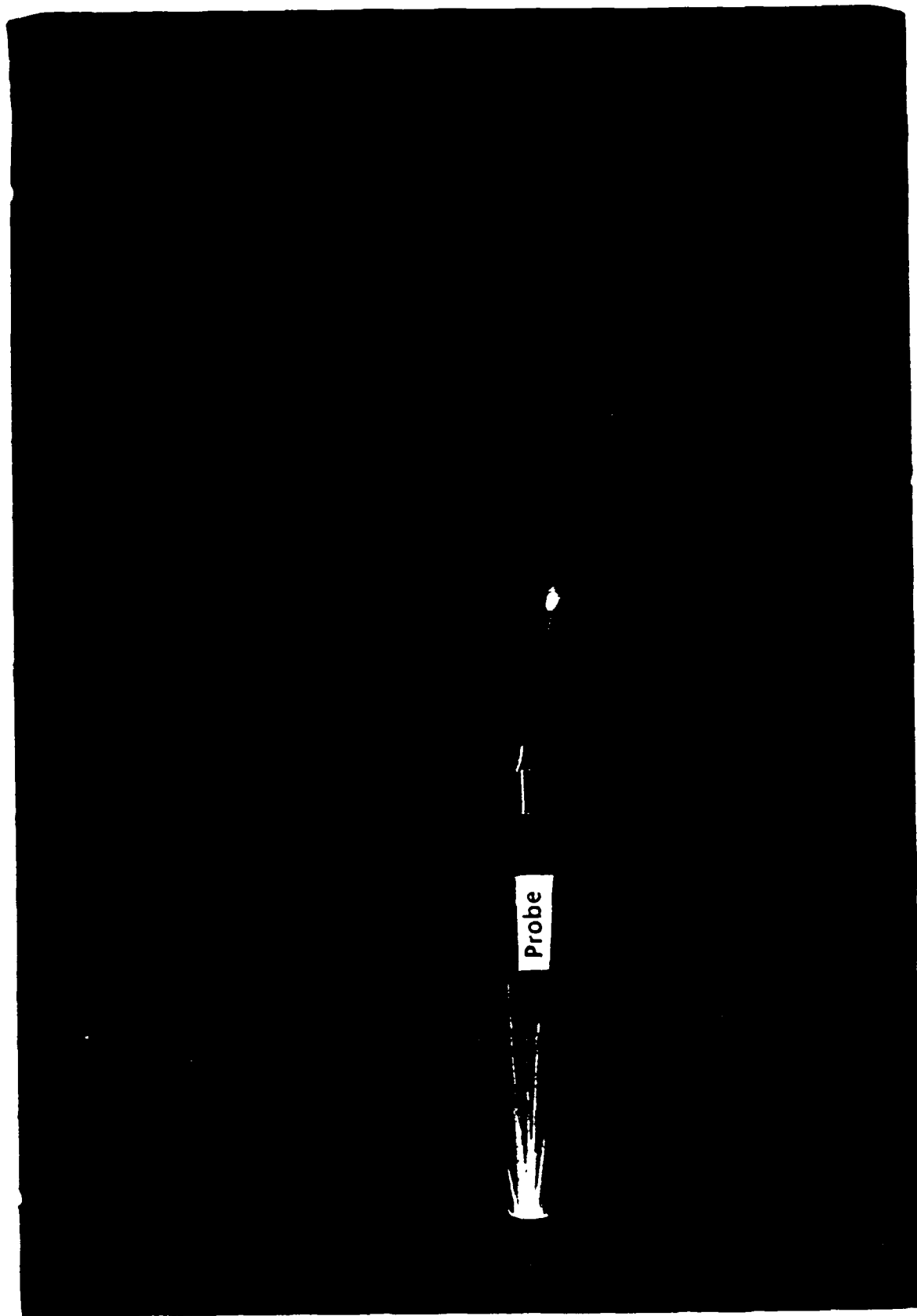


Figure 7. Probe during a test run.



Figure 8. Atomic oxygen generator and cooling tubing.

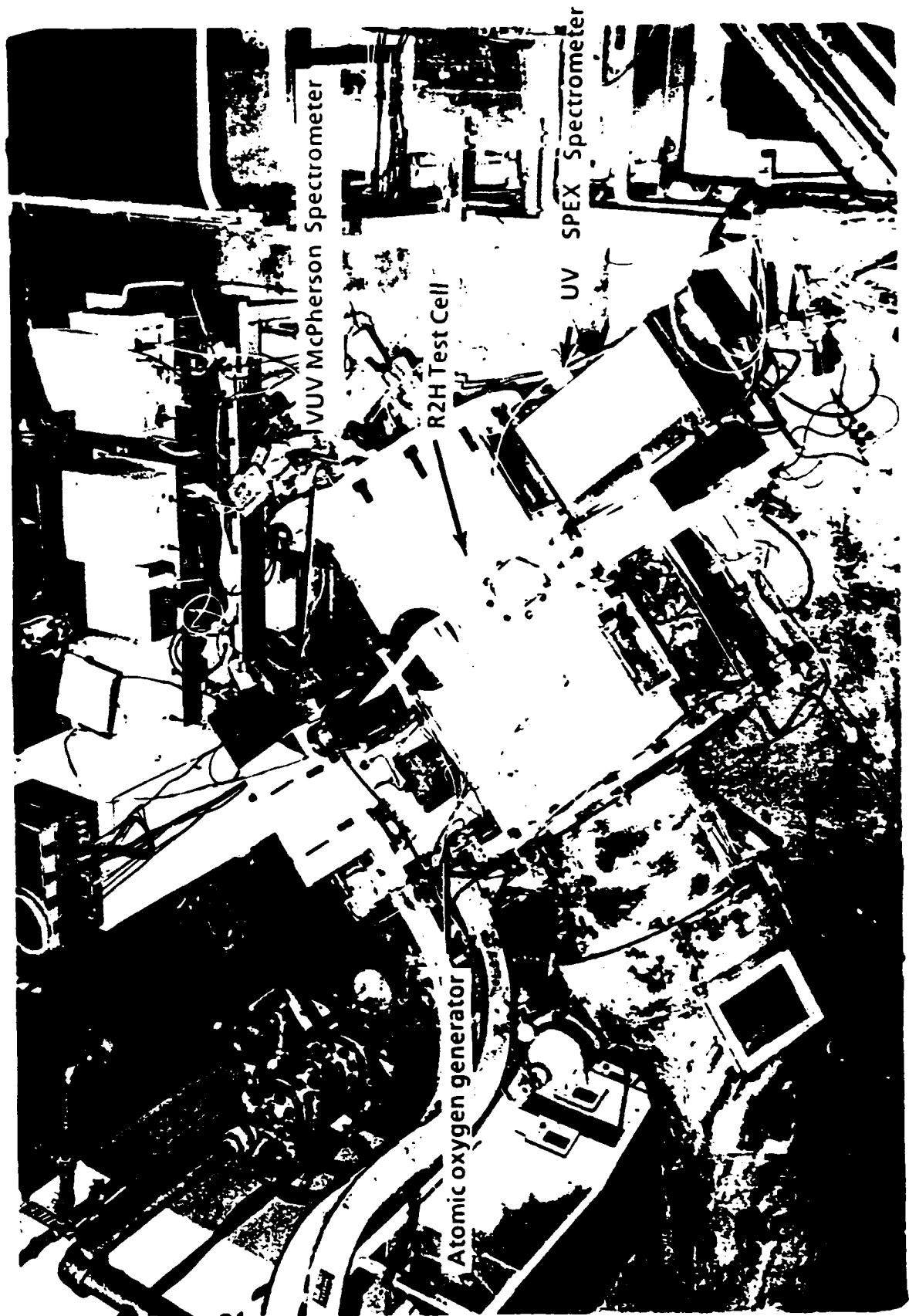


Figure 9. R2H test cell, two UV spectrometers, and atomic oxygen generator.

**ANALYSIS OF PROBLEMS IDENTIFIED IN PROJECT ENGINEER'S  
NOTEBOOKS IN THE VKF TUNNELS ABC**

John-Paul Motley  
Mentor: Jim Whoric  
Arnold Engineering Development Center

25 July 1991

## **ANALYSIS OF PROBLEMS IDENTIFIED IN PROJECT ENGINEER'S NOTEBOOKS**

Mr. John-Paul Motley

### **ABSTRACT**

Data was collected from project engineer's notes on problems that occurred in the VKF wind tunnels ABC before, during, and after testing during the 1990 and 1991 fiscal years. Many types of problems were identified, but one particular problem showed up in the majority of data. The computer communications network which links MACS, RADS, and the printers caused multiple delays which increased the total cost of the projects. Further investigation is needed on this data to acquire closer estimations on the magnitude of impact and the cost of the problem.

### **ACKNOWLEDGEMENTS**

First and foremost, I would like to thank Mr. Ray Eaves and Mr. Jim Whoric for having the confidence in me to perform this project. I would also like to thank all of the project engineer's in the Von Karman Facility Main Test and Lab Building for having the patience to work with me while I was collecting data for this project. Last, but certainly not least, I greatly thank Mr. Rick Beesley who has helped me through every step, every question, and every problem that arose. This project would have never been completed without his help. To everyone at the VKF building, thanks for a memorable experience!



## INTRODUCTION & DISCUSSION

During this study, data were collected from project engineer's notebooks on problems encountered during the various phases of test projects in wind tunnels A, B, and C. Each problem found was entered on a data base spreadsheet and categorized into different sections for ease of analysis(see Tables 1 and 2). Microsoft Excel was used to create the spreadsheet and Pareto charts. The data were used to develop a Pareto analysis of trouble resources as a guide to provide focus on making improvements in the overall operation in the ABC area. The data were also used to compare with the Trouble Reports data base to indicate where improvements should be made in the Trouble Reports process. The data that was found in both the Master Trouble Log and the project engineer's notebooks was eliminated, and a new spreadsheet containing only problems that had not been found in the Master Trouble Log was made(see Table 3). The spreadsheet was then mailed to Mr. Mel Homan of the Tunnel Operations Branch (4T/16T/S Operations Section) to be entered into the Wind Tunnel Problem Log. The two additional sections, Minor Facility Number and System Number, were requested by Mr. Homan for help in entering the new data. For this analysis, however, all of the data collected from the project engineer's notes is used. Even though all of this data is used, it is still important to note that from a statistical viewpoint, the amount of data

collected is very small, and therefore, it is impossible to do much detailed analysis. However, some overall observations can be made from the sections which the problems were divided. (Note: In case more detailed information is needed than the Pareto charts offered, the exact number of events and the exact cumulative percentage are given in Table 4 for each division in each section).

### RESULTS

A Pareto chart of the number of events by project phase is shown in Figure 1. It appears that most problems identified were during the testing and installation/check-out phases. This is to be expected, because the majority of the project engineer's notes were taken during these two phases. If further analysis could be done for the problems that occurred during testing, it should prove to be helpful. When a problem occurs during testing, it either reduces testing efficiency or results in downtime or delayed operations which increase the cost of the project.

In Figure 2, a Pareto chart of the number of events by individual problems is shown. This is where our conclusion is the most evident. The computer communications network which links RADS, MACS, and the printers accounts for a total of 37 of the 132 events or 28%. Lack of labor support is also a problem which might be further investigated along with signal conditioner malfunction. Both problems should be easily corrected.

Figure 3 is a Pareto chart of the number of events by functional group. In this figure, (1)Model Support & Control and (2)Data Acquisition & Analysis account for 70% of the problems. Many of the problems in both divisions are related to the computer communications network which seems to be the most evident problem in this data collection. Problems such as printer hang-ups, MACS and RADS dying, and replacements of signal conditioners are the reasons for these two divisions to be the highest.

Figure 4 shows a Pareto chart of the events by responsible organization. This chart is not a surprise either. The Instrumentation Systems Facility and Operations and Maintenance Facility account for 90% of the problems. ISF's main problems, of course, were the printers, RADS, and MACS. However, one of OMF's major problems was the lack of labor support. This is definitely an issue that can be worked on. OMF's other major problems consisted of troubles with probes, struts, and heaters.

A Pareto chart of the number of events by nature of impact is shown in Figure 5. Since the data was acquired from project engineer notebooks, it was often times difficult to categorize a problem. Some of the problems encountered did not directly delay operations; instead, the problems just made testing less efficient such as having to resort to manual inputs. One positive aspect of the data in

this chart is that they show little downtime indicating that the events that delay operations are more significant than those which cause downtime.

Presented in Figure 6 is a Pareto chart of the events by apparent cause of problem. Not surprisingly, the data indicate that equipment failures are the dominant problem. Such items as RADS and MACS dying and printers jamming are once again categorized in the top percentage group. However, personnel problems are significant. If the personnel divisions (Required Action Not Properly Taken, Required Action Not Taken in Timely Manner, and Required Personnel Not Available) are combined, they account for 36% of the events. Further analysis and investigation of this area should be considered in the future.

#### CONCLUSIONS

A study of problems encountered during the conduct of test projects in the Von Karman Facility Wind Tunnels A, B, and C during the 1990 and 1991 fiscal years as recorded in the project engineer's notebooks was conducted from June 17 to July 12, 1991. The major conclusions of the analysis are as follows:

1. Trouble with the computer communications network such as MACS, RADS, and the printers appear to be the most frequent problems individually.

2. Seventy percent of the problems were related to two functional groups: (1) Model Support & Control and (2) Data Acquisition & Analysis.

3. Equipment failure was identified as the number one cause of problems with personnel problems as the second major cause.

4. Thirty-six problems were recorded in the project engineer notebooks that had not been recorded in the Master Trouble Log indicating the need for closer coordination of trouble event documentation.

Many interesting facts about the ABC area were learned while working on this project. If it is costing \$50.00 per minute every time the communications network fails, it seems that an investigation of the problem could save much money in the cost of future projects. The Master Trouble Log and the Trouble Reports process seem to be of good use. If the Master Trouble Log is used to its potential and carefully studied, it may help the overall cost of the projects by creating more efficient work and causing less hours of delayed operations and downtime.

T A B L E 1

Section A - Facility (FAC)

- P - Propulsion Wind Tunnel Facility
- V - Von Karman Gas Dynamics Facility

Section B - Test Unit (TU)

- 4 - Supersonic Wind Tunnel A
- 5 - Hypersonic Wind Tunnel B
- 6 - Hypersonic Wind Tunnel C

Section I - Source

- OTL - Operation Time Log
- PEN - Project Engineer's Notebook

Section C - Phase (PH)

- 1 - Planning
- 2 - Design
- 3 - Fabrication
- 4 - Installation/Check-Out
- 5 - Testing
- 6 - Removal
- 8 - Reporting

Section J - Reported By  
(REPT BY)

- Project Engineer's Initials

Section D - Functional Group (FG)

- 1 - Process Air
- 2 - Model Support & Control
- 3 - Data Acquisition & Analysis
- 4 - Utilities & Support
- 5 - Test Article

Section E - Responsible Organization Code (ROC)

- ASF - Aerospace Systems Facility
- ISF - Instrumentation Systems Facility
- OMF - Operations and Maintenance Facility
- SPO - Sponsor
- SSI
- USER

Section F - Nature of Impact (NOI)

- 1 - Test Unit Downtime
- 2 - User Downtime
- 3 - Prevented Simultaneous Operation
- 4 - Unable to Schedule Operations
- 5 - Delayed Operations
- 6 - No Operational Impact

Section G - Magnitude of Impact (MOI)

Time of delay in hours, days, or weeks

Section H - Apparent Cause of Problem (ACP)

- 1 - Equipment Failure
- 2 - Required Action Not Taken in Timely Manner
- 3 - Required Action Not Properly Taken
- 4 - Required Personnel Not Available
- 5 - Developmental Problem
- 6 - Inadequate Design

TABLE 2

FAC.	TU	PR	DESCRIPTION OF PROBLEM	DATE	TIME	FG	ROC	NOI	MCI	ACP	SOURCE	REPT BY
V	4	4	Cut 0.040" tubes short and solder 0.060"	2/8/90		5	USER	6		3	PEN	PEF
V	4	4	Noise cooling system leaks about 2 drops per minute	10/4/90		5	USER	6		5	PEN	PEF
V	4	4	Unable to work on mass flow lines-safety restrictions	11/16/90		1	OMF	5	3.0 hrs	3	PEN	PEF
V	4	4	No crafts support-craftsmen loaned out to PWT	11/16/90		2	OMF	5		4	PEN	PEF
V	4	4	Denied access to installation tank	11/16/90		2	OMF	5		3	PEN	PEF
V	4	4	Venturi system valve leak-extra valve seat installed	11/16/90		1	OMF	5		3	PEN	PEF
V	4	4	MDC cut back hydraulic lines wrong	11/28/90		5	USER	5		3	PEN	PEF
V	4	4	Engine leaks from office to ESP	11/29/90	9:45	5	USER	5		1	PEN	PEF
V	4	4	Strut lock problems - pitch calibration delayed	12/3/90		2	OMF	5		1	PEN	PEF
V	4	4	Asbestos team working around tunnel A	12/12/90		4	OMF	6		2	PEN	PEF
V	4	4	Weak power supply - have to change	3/7/91	4:35	4	ISF	6		3	PEN	CLR
V	4	4	Signal conditioner malfunction - had to replace	3/7/91	5:00	2	ISF	5	7.5 hrs	1	PEN	CLR
V	4	5	Fin Z/4 disconnects during model change	3/8/91	1:54	5	OMF	5	1.5 hrs	6	PEN	CLR
V	4	5	Fin balance wires broken - have to solder	3/11/91	5:15	5	OMF	6	0.6 hr	6	PEN	CLR
V	4	5	No data printing out	3/12/91	0:46	3	ISF	6		1	PEN	CLR
V	4	5	Unable to open safety doors - air off	3/12/91	4:09	4	OMF	6	18.0 hrs	5	PEN	CLR
V	4	5	Plant charging high pressure air - unable to air on	3/13/91	0:00	1	ASF	5	1.0 hr	2	PEN	CLR
V	4	4	MACS error - drive privilege not available	3/13/91	23:05	2	ISF	6		3	PEN	CLR
V	4	4	Strut problems - locking and unlocking	3/15/91	1:00	2	OMF	6		1	PEN	CLR
V	4	5	Difficulties in nozzle control	3/17/91	0:00	1	OMF	6	0.25 hr	1	PEN	CLR
V	4	5	Suck down taking too long	3/19/91	0:52	1	OMF	5		2	PEN	CLR
V	4	5	Printer hang-up	3/19/91	1:00	3	ISF	6		1	PEN	CLR
V	4	5	Printer hang-up	3/19/91	2:29	3	ISF	6		1	PEN	CLR
V	4	5	Printer hang-up	3/20/91	4:41	3	ISF	6		1	PEN	CLR
V	4	5	Plots in textile are all "wiped out"	3/20/91	23:45	3	ASF	6		3	PEN	CLR
V	4	5	Printer hang-up	3/21/91	0:25	3	ISF	5		1	PEN	CLR
V	4	5	MACS malfunction after run	3/21/91	1:53	2	ISF	6		1	PEN	CLR
V	4	5	Printer hang-up	3/21/91	3:03	3	ISF	6		1	PEN	CLR
V	4	5	Back-up printer malfunctions	3/21/91	3:23	3	ISF	6		1	PEN	CLR
V	4	5	Dropped fin tab #2 during model change	3/21/91	3:45	5	OMF	6		3	PEN	CLR
V	4	5	MACS malfunction	3/21/91	7:25	2	ISF	6		1	PEN	CLR
V	4	5	Signal conditioner making too much noise	3/22/91	4:31	2	ISF	6		3	PEN	CLR
V	4	5	Main printer malfunction	3/25/91	1:35	3	ISF	6		1	PEN	CLR
V	4	5	RADS did not pick up run	3/25/91	4:02	3	ISF	5		1	PEN	CLR
V	4	1	Model shear pin redesigned	3/25/91		5	USER	5		6	PEN	JPG
V	4	4	Druck leads - too short	4/3/91		2	OMF	5	3.0 hrs	6	PEN	SMC
V	4	4	Lack of support in second shift	4/3/91		2	ISF	4	4.0 hrs	4	PEN	SMC
V	4	5	Unable to remove coated screws	4/4/91		2	USER	5		5	PEN	SMC
V	4	1	Working computer program late	4/9/91		3	ISF	4	4 wks	3	PEN	JPG
V	4	3	Water cut off - vax unable to operate	4/9/91		4	OMF	5	2 days	2	PEN	JPG
V	4	4	Balance to spinning screw plugged	4/9/91		2	OMF	5	4.0 hrs	3	PEN	JPG
V	4	6	PE not named at base pressure tube leak	4/9/91		3	ISF	6		3	PEN	SMC

TABLE 2 (CONT.)

FAC	TU	PH	DESCRIPTION OF PROBLEM	DATE	TIME	FG	RUC	MOT	MOT	ACP	SOURCE	REPT BY
V	4	6	PBI open-affected reference-took atmospheric zeros	4/9/91		3	ISF	6		1	PEN	SMC
V	4	4	Autospin controller malfunction during lab checkout	4/23/91		2	ISF	6		5	PEN	JPG
V	4	4	Both printers in control room stop functioning	4/25/91	2:20	3	ISF	6		1	PEN	JPG
V	4	4	Autospin controller malfunction	4/25/91	6:30	2	ISF	6		5	PEN	JPG
V	4	4	Tunnel A nozzle malfunction	4/26/91	1:15	1	OMF	5	8.0 hrs	1	PEN	JPG
V	4	4	First tare check unacceptable - no explanation	4/27/91	13:24	3	ISF	6		1	PEN	JPG
V	4	5	Plant controlling computer failed	4/28/91	20:00	4	OMF	1	8.0 hrs	1	PEN	JPG
V	4	5	Model roll pin sheared during model change	4/29/91	23:15	2	OMF	6		6	PEN	JPG
V	4	5	RADS / MACS dies eleven times during shift	4/30/91	2:00	2	ISF	6	2.0 hrs	1	PEN	JPG
V	4	5	Unable to reverse spin	4/30/91		2	ASF	6		6	PEN	JPG
V	4	4	MACS malfunction - reads "Read a loop of data"	5/14/91	4:00	2	ISF	6		1	PEN	DH
V	4	6	Wrench broke off in screw	5/15/91	2:55	6	OMF	5	0.2 hr	6	PEN	DH
V	4	5	Jack would not pull out properly	5/15/91	4:00	4	OMF	5	0.2 hr	1	PEN	DH
V	4	5	Encountered electrical limit on roll - had to override	5/15/91	4:34	2	OMF	8		1	PEN	DH
V	4	5	Valve problem preventing tank shutdown	5/15/91	6:50	1	OMF	5	0.07 hr	1	PEN	DH
V	4	5	MACS malfunction - look two runs manually	5/15/91	7:10	2	ISF	6		1	PEN	DH
V	4	5	Shing overload - foul did not show up in lab data	5/16/91	1:15	3	ISF	6		1	PEN	DH
V	4	5	Had to reconnect MACS to network	5/16/91	2:22	2	ISF	6		1	PEN	DH
V	4	5	Small part omitted from model config - data repeated	5/17/91	2:20	3	ASF	5	0.1 hr	3	PEN	DH
V	4	6	Signal conditioner replaced	5/17/91	2:50	3	ISF	6	0.1 hr	1	PEN	DH
V	4	5	Tunnel would not run at required condition	5/17/91	7:30	4	OMF	1	1.5 hrs	1	PEN	DH
V	4	5	Plant overheating on bearing of a compressor	5/17/91	0:20	2	ISF	1	1.5 hr	1	PEN	DH
V	4	5	Computer program not storing zeros right	5/17/91	2:00	3	ISF	6		2	PEN	DH
V	4	5	Dew point instrument malfunction	5/17/91	5:20	1	OMF	5		1	PEN	DH
V	4	5	Final oruck zero impossible-valve is heating the drums	5/17/91	1:00	3	ISF	6		6	PEN	DH
V	4	5	Rolling caused foul wire to pull back and contact shing	5/17/91	0:35	2	OMF	1	0.5 hr	3	PEN	DH
V	4	5	Wrong configuration run	5/17/91	3:20	5	ASF	2	0.1 hr	3	PEN	DH
V	4	5	Foul strip gives erroneous constant foul signal	5/17/91	5:20	2	OMF	6		1	PEN	DH
V	5	4	Craftsmen in meeting-unavailable for weight hanging	12/11/90	13:05	2	OMF	5	1.0 hr	4	PEN	CLR
V	5	4	RADS 1 data loop period lengthened due to RADS 2	12/12/90	9:45	3	ISF	6		1	PEN	CLR
V	5	4	Controller problems with roll relay switch	12/13/90	13:35	2	ISF	6		1	PEN	CLR
V	5	5	Controller problem with model roll again	12/13/90	17:35	2	ISF	6		1	PEN	CLR
V	5	5	Printer malfunction	12/14/90	16:50	3	ISF	6		1	PEN	CLR
V	5	5	Plant lost heater	12/14/90	16:50	4	OMF	6		1	PEN	CLR
V	5	5	Speed controller only rolls negative	12/14/90	23:25	2	ISF	5	0.1 hr	1	PEN	CLR
V	5	5	Window insert set screw broke off inside model	3/4/91	0:15	5	OMF	1	8.0 hrs	5	PEN	KWN
V	5	4	Program not storing zeros correctly	4/18/91	11:15	3	ISF	6		3	PEN	MGH
V	5	5	Dew point will not drop	4/17/91	0:35	1	OMF	5	0.73 hr	1	PEN	MGH
V	5	5	RADS died	4/17/91	2:10	3	ISF	5	0.43 hr	1	PEN	MGH
V	5	4	Flow switch not working on water system	5/6/91	0:30	4	OMF	6		1	PEN	JCD
V	5	4	Probes would not retract-ran into model surface	5/6/91	5:00	2	ISF	5	13.0 hrs	1	PEN	JCD
V	5	5	House air not sufficient to raise shield under air loads	5/8/91	1:05	1	OMF	6		1	PEN	JCD



TABLE 2 (CONT.)

FAC	TU	PH	DESCRIPTION OF PROBLEM	DATE	TIME	FG	ROC	MOI	ACP	SOURCE	REPT BY	
V	6	4	ESP reference inadequate	6/20/90		2	ISF	5	1	PEN	PEF	
V	6	4	ESP problems-leak at calibrate point	6/21/90		2	ISF	5	1	PEN	PEF	
V	6	4	SPS UNKNOWN inaccurate - improper zeroing	6/22/90		3	ISF	6	3	PEN	PEF	
V	6	4	RJ Shifted	6/22/90		3	ISF	6	1	PEN	PEF	
V	6	5	Flow not passing through HB-1 heater	6/25/90	0:47	1	OMF	5	23.5 hrs	1	PEN	PEF
V	6	5	RJ jumps 300 counts - caused by condensation?	6/25/90	1:02	3	ISF	6	1	PEN	PEF	
V	6	5	Centerline indicate not received	6/26/90	4:01	2	OMF	6	1	PEN	PEF	
V	6	5	L-5 firing stops plant cranking	6/26/90	23:02	4	OMF	3	1.5 hrs	0	PEN	PEF
V	6	5	Heat exchanger water leak saturated DP instrument	6/27/90	2:42	3	OMF	6	1	PEN	PEF	
V	6	5	Centerline not registering - sector out of parallel	6/27/90	7:35	2	OMF	5	0.1 hr	1	PEN	PEF
V	6	5	Lost heater	6/27/90	10:15	1	OMF	5	0.5 hr	1	PEN	PEF
V	6	4	MFA tubes plugged - silver solder cracked	6/28/90		2	OMF	8	3	PEN	PEF	
V	6	4	MACS and controller not communicating	6/29/90		2	ISF	6	3	PEN	PEF	
V	6	4	Probe foot leak	7/2/90		2	ISF	5	1	PEN	PEF	
V	6	4	TT probe damaged from time it left shop and time installed	7/2/90		3	ISF	6	3	PEN	PEF	
V	6	4	RADS problem - negative manual input unrecognized	7/2/90		3	ISF	5	1	PEN	PEF	
V	6	4	MFA probe leak - silver solder cracked	7/2/90		2	ISF	5	1	PEN	PEF	
V	6	5	TT probe - junction improperly positioned	7/3/90		2	ISF	5	3	PEN	PEF	
V	6	5	Brittle silver solder cracked after pilot probe was installed	7/4/90		2	ISF	5	1	PEN	PEF	
V	6	5	Shift cancelled due to lack of support	7/6/90		2	OMF	4	8.0 hrs	4	PEN	PEF
V	6	4	AF project manager not notified immediately of add. surveys	7/6/90		3	ASF	6	2	PEN	PEF	
V	6	5	Lube oil in heat exchanger not flowing properly	7/10/90		4	OMF	6	3	PEN	PEF	
V	6	4	Check problem delivered with test peculiar parameters=0.000	7/3/91	14:00	3	ISF	6	3	PEN	SMC	
V	6	4	SSI or OPS Center failed to schedule guards	7/8/91	7:00	4	SSI	6	2	PEN	SMC	
V	6	4	Could not calibrate pitch and roll	7/8/91	9:00	2	ISF	6	5	PEN	SMC	
V	5	4	Photographers unable to mount camera - no harness	7/9/91		2	OMF	6	2	PEN	SMC	
V	6	4	Check problem - chosen data inadequate for curve fit	7/9/91		3	ISF	6	3	PEN	SMC	
V	6	4	Cooling water applied to ESP reference line - ESP destroyed	7/10/91	14:39	4	OMF	5	1 day	3	PEN	SMC
V	6	4	ESP buildup delayed by personnel conflict	7/11/91		2	ISF	5	3	PEN	SMC	
V	6	4	Pitch readout jumping up and down - W. Ward fixes	7/15/91	9:03	3	ISF	6	1	PEN	FWB	
V	6	4	Problem in balance wiring - detected by Ed Solomon	7/16/91	10:30	2	OMF	6	3	PEN	SMC	
V	6	4	Cooling water leak from inline roll mechanism - fixed itself	7/17/91	23:40	2	OMF	6	6	PEN	FWB	
V	6	5	Air leak through instrument ring	7/18/91	2:17	1	OMF	1	5.0 hrs	3	PEN	FWB
V	6	5	Problem with data system - "16 bit"	7/21/91	10:00	3	ISF	6	1	PEN	FWB	
V	6	5	Spikes in ESP data	7/21/91	11:00	3	ISF	6	1	PEN	FWB	
V	6	5	MACS problem preventing angle checks	7/21/91	23:55	2	ISF	5	0.5 hr	1	PEN	FWB
V	6	5	Problem with data system - "16 bit"	7/22/91	0:05	3	ISF	6	1	PEN	FWB	
V	5	5	Lost gas heater	7/22/91	2:17	1	OMF	1	0.5 hr	3	PEN	FWB
V	6	5	Cooling water leak from inline roll mechanism	7/22/91	1:06	2	OMF	6	6	PEN	FWB	
V	6	5	Problem with data system - "16 bit"	7/22/91	1:30	3	ISF	6	1	PEN	FWB	
V	6	5	Downlock not removed - could not inject	7/22/91	2:00	2	OMF	5	0.08 hr	3	PEN	FWB
V	6	5	Program error - program ignoring first digit of manual input	7/22/91	6:00	3	ISF	6	3	PEN	FWB	

T A B L E 2 (CONT.)

FAC	TU	PH	DESCRIPTION OF PROBLEM	DATE	TIME	FG	ROC	NOI	MOI	ACP	SOURCE	REPT BY
V	6	5	Data problems with PTANK - "on wrong port"	1/25/91	1:10	3	ISF	5		3	PEN	FWS
V	6	6	Faulty digital inclinometer	2/2/91		2	ISF	5	3.0 hrs	1	PEN	FWS
V	5	5	Flange leaking at plant - have to tighten - air off	5/29/91	9:52	1	OMF	5	2.1 hrs	2	PEN	CLR
V	6	5	Lost ST drive - have to restage	5/29/91	12:53	1	OMF	5	0.37 hr	1	PEN	CLR
V	6	5	Problems with venting tank to atmosphere	5/29/91	13:15	1	OMF	5	0.07 hr	1	PEN	CLR
V	6	5	Water leak	5/29/91	15:23	2	OMF	6		1	PEN	CLR
V	7	5	Masscomp lost data	11/2/90	18:38	3	ICI	5	3.0 hrs	3	PEN	EJM

TABLE 3

TEST FAC UNIT	PM	DESCRIPTION OF PROBLEM	DATE OCC'D	TIME OCC'D	MINOR FAC. NUMBER	FUNC. GROUP	RESP. ORG	NATURE OF IMPACT	MAG. OF IMPACT	APP. CAUSE OF PROBLEM	SYSTEM NUMBER	SOURCE OF INPUT	REPAIR BY
V	4	Weak power supply - have to change	3/7/91	4:35	14	4	ISF	5		3	2CB-IC1-00676-160302	PEN	CLP
V	4	Signal conditioner malfunction	3/7/91	5:00	9	7	ISF	5	7.5 hr	1	2CB-IC1-00676-160302	PEN	CLP
V	4	Fin balance wires broken - have to solder	3/7/91	5:15	9	5	OMF	5	0.6 hr	6		PEN	CLP
V	4	No data printing out	3/12/91	0:45	9	3	ISF	6		1	2C-IB2-00676-1642	PEN	CLP
V	4	Plant charging high pressure air	3/13/91	0:00	14	1	ASF	5	1.0 hr	2	2C-SD1-00065-0202	PEN	CLP
V	4	MACS error - drive privilege not available	3/13/91	23:05	9	2	ISF	6		3	2CB-IB3-00676-090105	PEN	CLP
V	4	Suck down taking too long	3/19/91	0:52	9	1	OMF	5		2	2C-SB4-00676-4001	PEN	CLP
V	4	Printer hang-up	3/19/91	2:29	0	3	ISF	5		1	2C-IB2-00676-1642	PEN	CLP
V	4	Man's in telexie are all "wiped out"	3/20/91	23:45	9	3	ASF	6		3		PEN	CLP
V	4	Printer hang-up	3/21/91	3:02	0	3	ISF	6		1	2C-IB2-00676-1642	PEN	CLP
V	4	Back-up printer malfunctions	3/21/91	3:23	9	3	ISF	6		1	2C-IB2-00676-1643	PEN	CLP
V	4	Dropped fin tab #2 during model change	3/21/91	3:45	5	5	OMF	6		3		PEN	CLP
V	4	Signal conditioner making too much noise	3/22/91	4:31	9	2	ISF	5		3	2CB-IC1-00676-160302	PEN	CLP
V	4	Lack of support in second shift	4/3/91		9	2	ISF	4	4.0 hrs	4		PEN	SMC
V	4	Working computer program: late	4/9/91		0	2	ISF	4	4 wks	3	2CC-IB3-00676-050403	PEN	JPG
V	4	Water cut on - vox unable to operate	4/9/91		14	4	OMF	5	2 days	2	2CB-IB2-00676-0504	PEN	JPG
V	4	Balance to spin rig screw plugged	4/9/91		9	2	OMF	5	4.0 hrs	3	2S-SB4-00676-3806	PEN	JPG
V	4	PE not noticed of base pressure tube leak	4/9/91		9	3	ISF	6		3		PEN	SMC
V	4	PBI open - had to take atmospheric zeros	4/9/91		9	3	ISF	6		1		PEN	SMC
V	4	Autospin controller malfunction	4/23/91		9	2	ISF	6		5		PEN	JPG
V	4	First late check unacceptable	4/27/91	13:24	9	3	ISF	6		1	2CB-IC1-00676-1603	PEN	JPG
V	4	MACS malfunction	5/14/91	4:00	9	2	ISF	6		1	2CB-IB4-00676-0901	PEN	DH
V	4	Wrench broke off in screw	5/15/91	2:55	9	5	OMF	5	0.2 hr	6		PEN	DH
V	4	Encountered electrical limit on roll	5/15/91	4:34	9	2	OMF	8		1	2C-SB4-00676-3822	PEN	DH
V	4	MACS malfunction - took two runs manually	5/15/91	7:10	9	2	ISF	5		1	2CB-IB4-00676-0901	PEN	DH
V	4	Had to reconnect MACS to network	5/16/91	2:22	9	2	ISF	6		1	2CB-IB3-00676-090105	PEN	DH
V	4	Seal port omitted from model config	5/17/91	2:20	9	3	ASF	5	0.1 hr	3		PEN	DH
V	4	Signal conditioner replaced	5/17/91	2:50	9	3	ISF	5	0.1 hr	1	2CB-IC1-00676-160302	PEN	DH
V	4	Tunnel would not run at required condition	5/17/91	7:30	9	4	OMF	1	1.5 hrs	1	2-30001-0393	PEN	DH
V	4	Computer program not during zeros night	5/17/91	2:00	9	3	ISF	6		2	2CC-IB3-00676-050403	PEN	DH
V	4	Dew point instrument malfunction	6/12/91	6:20	9	1	OMF	6		1	2CB-DI1-00676-1612	PEN	DH
V	4	Final druck zero impossible	6/13/91	1:00	9	3	ISF	6		6		PEN	DH
V	4	Wrong configuration run	5/14/91	3:29	9	5	ASF	2	0.1 hr	3		PEN	DH
V	4	Program not during zeros correctly	4/16/91	11:15	10	3	ISF	6		3	2CC-IB3-00676-050403	PEN	MGH
V	4	Dew point will not drop	4/17/91	0:35	10	1	OMF	5	0.73 hr	1	2CB-DI1-00676-1612	PEN	MGH
V	4	RAOS died	4/17/91	2:19	10	3	ISF	5	0.43 hr	1	2CB-IC1-00676-1603	PEN	MGH

T A B L E    4

<u>Section</u>	<u>Division</u>	<u># of</u>	<u>Cum. %</u>
<u>Phase</u>		<u>Events</u>	
	Testing	77	57.9
	Installation/Check-Out	50	95.5
	Removal	3	97.7
	Planning	2	99.2
	Fabrication	1	100.0
	Design	0	100.0
	Reporting	0	100.0
Functional Group	Model Support & Control	51	38.3
	Data Acquisition & Analysis	41	69.2
	Process Air	17	82.0
	Utilities & Support	13	91.7
	Test Article	11	100.0
Responsible Org. Code	ISF	65	49.2
	OMF	54	90.2
	ASF	6	94.7
	USER	6	99.2
	SSI	1	100.0
	SPO	0	100.0
Nature of Impact	No Operational Impact	72	54.1
	Delayed Operations	49	91.0
	Test Unit Downtime	7	96.2
	Unable to Schedule Operations	3	98.5
	User Downtime	1	99.2
	Prevented Simultaneous Operation	1	100.0
Apparent Cause of Problem	Equipment Failure	68	51.5
	Req. Action Not Properly Taken	34	77.3
	Inadequate Design	12	86.4
	Req. Act. Not Taken in Timely Manner	9	93.2
	Developmental Problem	5	97.0
	Required Personnel Not Available	4	100.0

FIGURE 1 - PROBLEMS BY PHASE

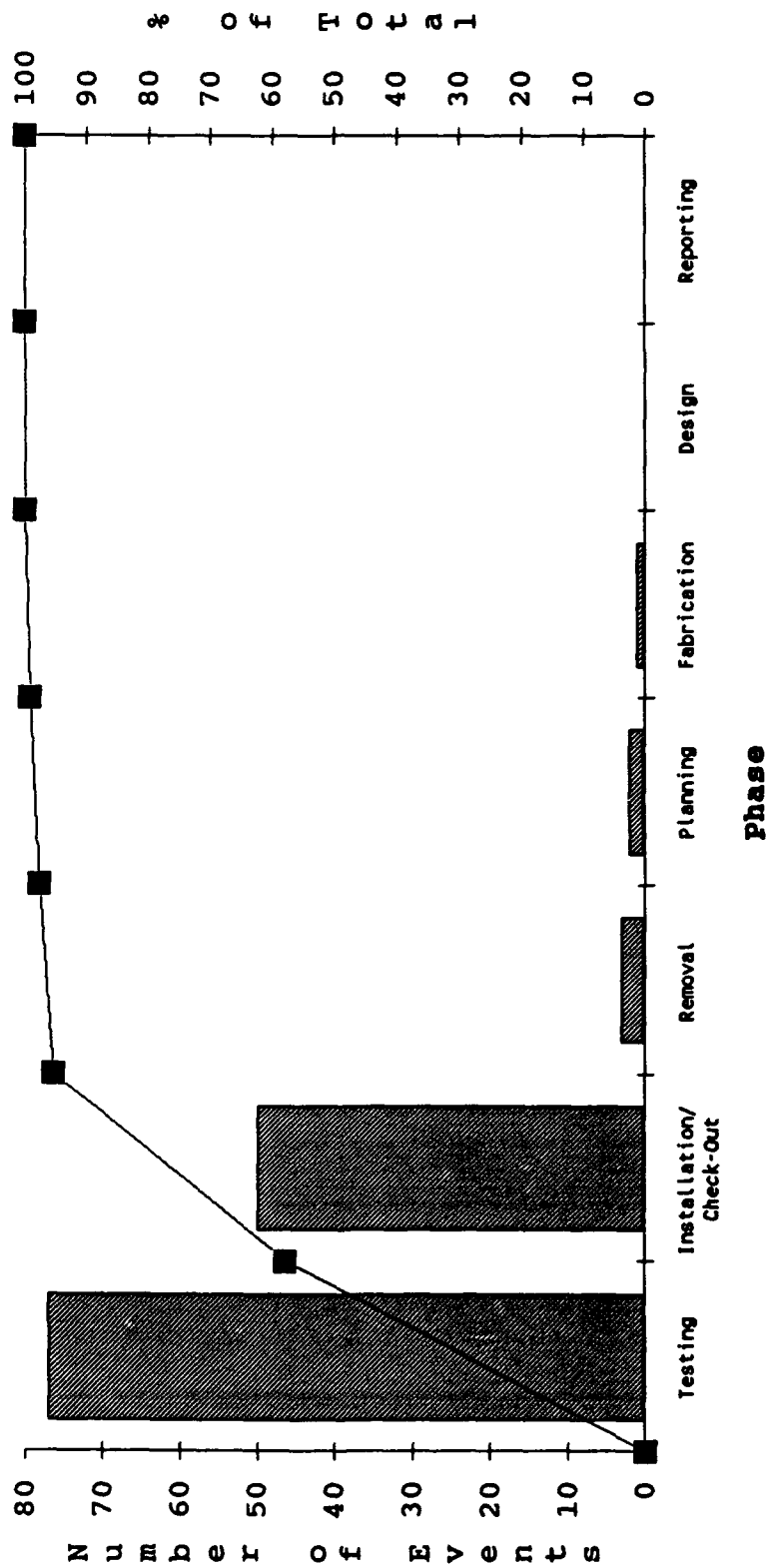


FIGURE 2 - INDIVIDUAL PROBLEMS

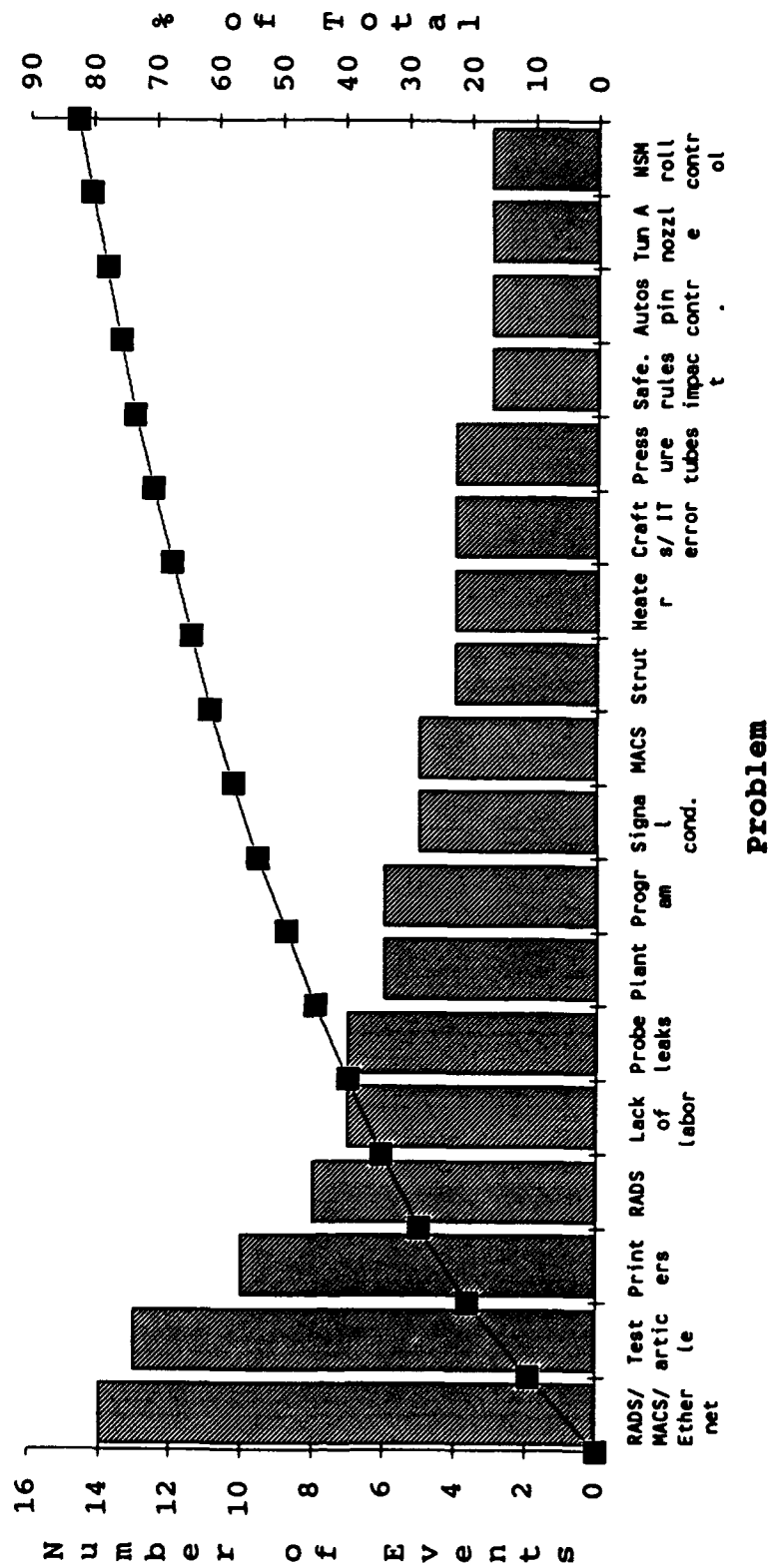


FIGURE 3 - PROBLEMS BY FUNCTIONAL GROUP

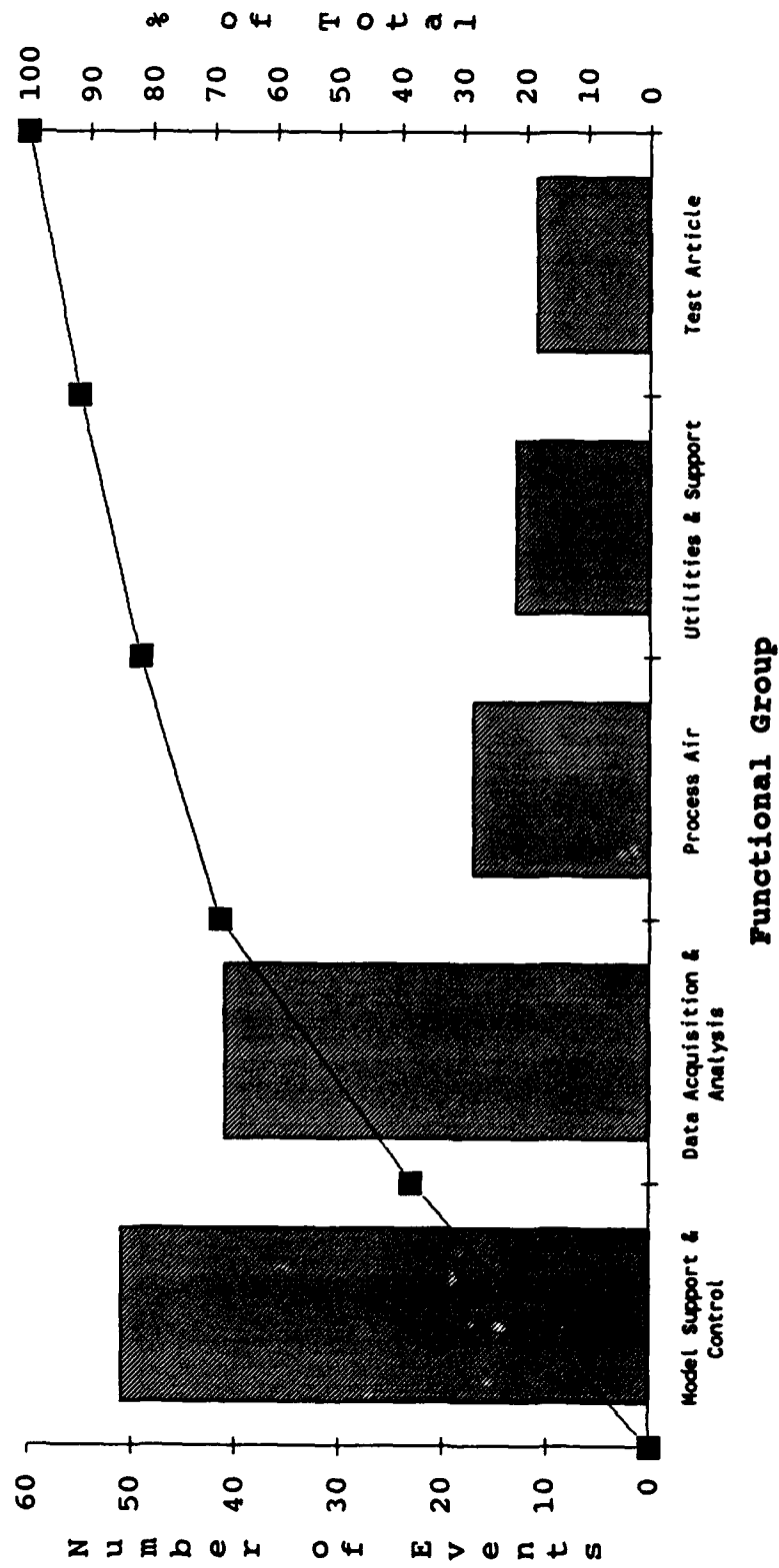


FIGURE 4 - PROBLEMS BY RESPONSIBLE ORGANIZATION

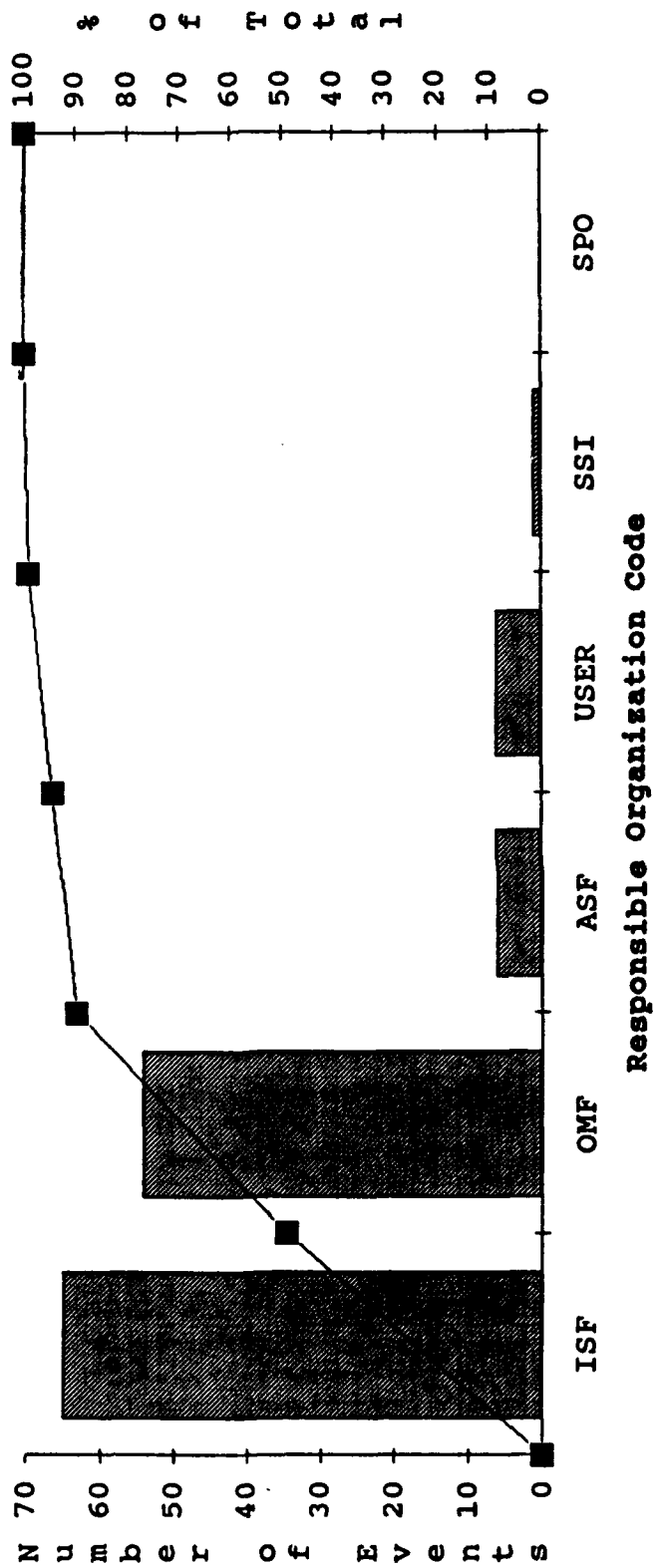




FIGURE 5 - IMPACT OF PROBLEMS

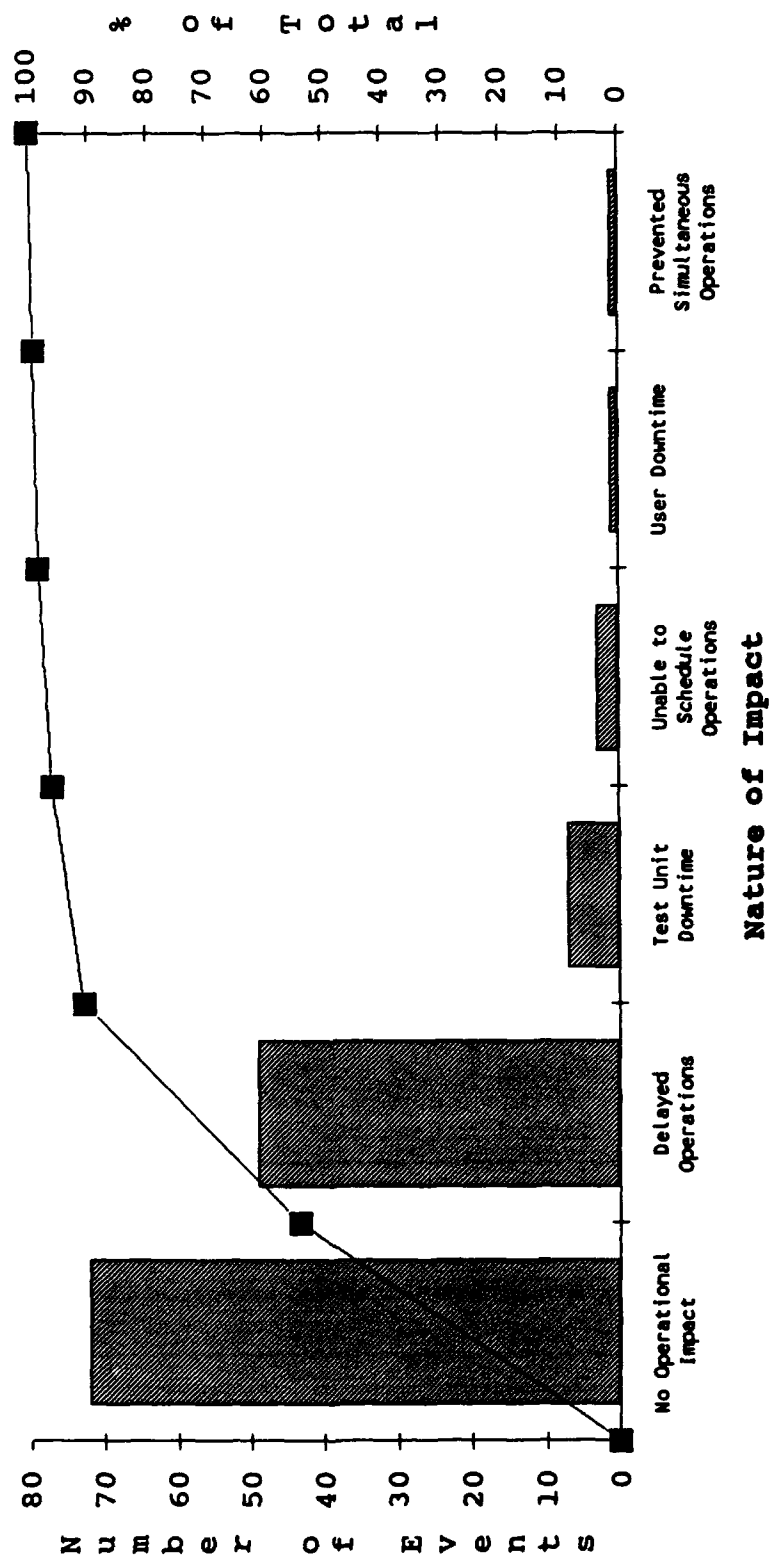
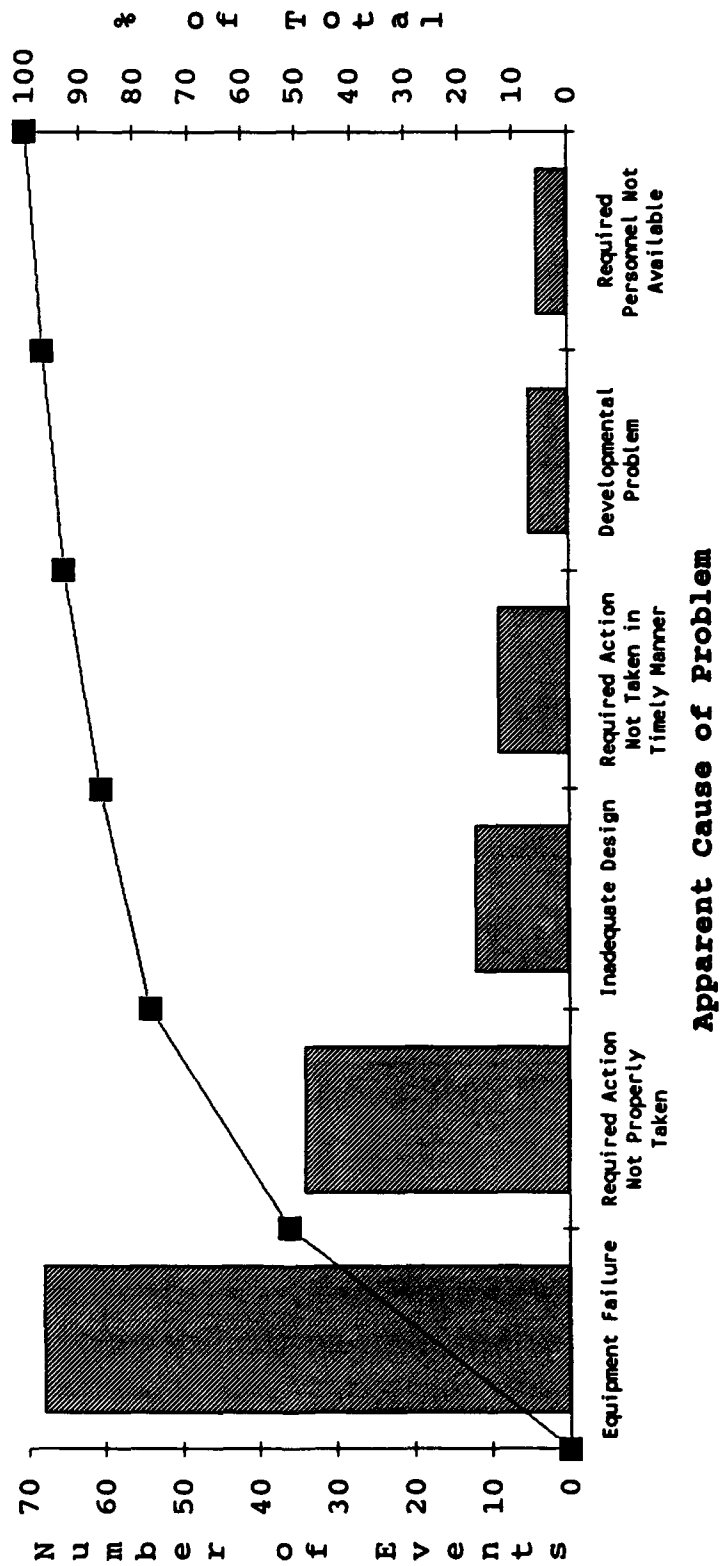


FIGURE 6 - CAUSE OF PROBLEMS



**A THEORETIC TECHNIQUE FOR ASSESSING COMBUSTION  
INSTABILITY OF A LIQUID ROCKET ENGINE**

**Julie Reece**

**Mentor: Max Roler, Sverdrup Technology, Inc.**

**AEDC**

**August 9, 1991**

**1.0 ABSTRACT**

The objective of this study is to determine the sensitivity of liquid rocket engine combustion instability to injector spray characteristics and combustor performance. A newly developed computer code, ROCCID, was applied to a specified test article and parametric variations were performed. This determination would lead to minimizing the probability of combustion instability in liquid rocket engines. Droplet size model, propellant supply temperature, and mixing efficiency concluded to have the greatest effect on combustion instabilities. Future studies to determine the sensitivity of injector type and damping devices

to combustion stability should be performed.

## **2.0 ACKNOWLEDGEMENTS**

Appreciation is due to the Air Force Office of Scientific Research and Research & Development Laboratories for providing the opportunity to participate in the High School Apprenticeship Program. I am grateful to Max Roler for providing specific opportunities to learn about rocket propulsion. Appreciation is also addressed to Rob McAmis for his time and guidance.

## **3.0 INTRODUCTION**

AEDC is responsible for providing the propulsion community with simulated altitude capability for all types of propulsion systems. Ground testing of development or qualified liquid rocket engines is critical to assess system performance. Historically, combustion instabilities have been investigated through rigorous experimentation. To supplement experimentation and provide foresight to combustion instability, analytical codes are required. In liquid rockets, combustion is never ideal. That is, fluctuations of pressure, temperature, and velocity are always present. Combustion instability can occur when these fluctuations combine with the natural frequencies of the propellant feed system or the

chamber acoustics (Ref. 1). Classification of combustion instabilities range from low to high frequencies, where high frequency instabilities are the most destructive. Combustion instabilities may result in uncontrolled impulse, extensive damage to the thrust chamber, extensive damage to injector, and decreased performance (Ref. 2). Experimental studies of combustion instabilities in liquid rocket engines are quite costly and difficult because scale model testing of instabilities are almost impossible, and full scale model testing of instabilities is costly. The Rocket Combuster Interactive Design (ROCCID) code addresses this complex problem of liquid rocket engine combustion instability experimentation. ROCCID is empirically formulated and is subsequently described in detail in section 5.1.

A parametric study to determine the sensitivity of certain operating parameters was performed. The ROCCID code was used to compute steady-state performance, low frequency, and high frequency instabilities.

#### **4.0 EXPERIMENTAL TEST DATA**

To evaluate the code, an experimental test case was used. An experimental liquid rocket engine designed and tested by Aerojet Corporation was chosen for this study (Ref. 3). Combustion chamber geometry, injector design, and propellants were kept constant throughout the study. The propellant

combination included liquid oxygen as the oxidizer and RP-1 as the fuel. The injector design consisted of 105 O-F-O triplet elements shown in Figures 1 and 2. The cylindrical chamber geometry lacked any baffles or acoustic devices, Figure 3. Chamber pressure, droplet size, mixing efficiency, propellant temperature, O/F ratio, burning response model, and burning response time constant were the operating conditions which were varied. The defined base line case has the following operating conditions: a chamber pressure of 1250 psi, the Aerojet droplet size model, the Aerojet burning response time constant, the combustion response program (CRP) burning response model, a mixing efficiency of 1.00, an oxidizer supply temperature of -260°F, a fuel supply temperature of 77°F, and an O/F ration of 2.8.

## 5.0 APPROACH

### 5.1 Code Description

The Rocket Combuster Interactive Design (ROCCID) program, which was finalized in May of 1991, was created by linking the most advanced performance and combustion stability models into one code (Ref. 4). ROCCID's structure consists of three main components, which are: an interactive front end (IFE), which includes input, output, and replay files; a point analysis option, which provides performance analysis; and a point design

option, which creates the combustor design features from specified design requirements. This study used the Point Analysis option. Propellant atomization, vaporization, and mixing are included in the steady state combustion analysis. Four empirical models for propellant droplet size are included to choose from; Aerojet, Priem, Dropmix, and the swirl coaxial correlation (UTRC).

The combustor's performance is defined by the characteristic exhaust velocity ( $C^*$ ) and specific impulse (ISP)-based energy release efficiencies (ERE). These values account for combustion efficiency losses resulting from incomplete vaporization and/or mixing. The effects of acoustic damping devices, baffles and helmholtz resonators, are also computed by ROCCID.

An interactive front end allows the user to generate input, create files, and display output. All previous case inputs are stored as replay files and are available for editing. Combustion gas property tables are generated within ROCCID using one dimension kinetics and are also stored. ROCCID is operational on VAX 8600 series computer.

## 5.2 Code Application

The chamber geometry, injector design, and propellant combination were kept constant throughout the study. Droplet size, burning response model, burning response time

constant, mixing efficiency, propellant supply temperature, and O/F ratio were the operating parameters that were varied for the high frequency combustion stability analysis. The droplet size, mixing efficiency, and O/F ratio were varied for the low frequency combustion stability analysis. Tables 1 and 2 provide the complete listing of parametric test cases. Included in this table is maximum amplitude of combustion instability. An amplitude greater than one indicates that sufficient energy is available for the fluid processes to couple with the combustion processes and combustion instabilities to occur.

## 6.0 RESULTS

Plots were generated for all test cases. Examples of types of plots that were produced are included in Figures 4, 5, and 6. After reviewing results produced by ROCCID, droplet size model and mixing efficiency were found to have the greatest affect on combustion instability maximum amplitude. Propellant supply temperature, O/F ratio, and pressure had an insignificant influence on combustion instability maximum amplitude.

Figure 7.a presents the variance in maximum amplitude of high frequency combustion instability as a function of empirical droplet size model and O/F ratio. Larger RP-1 droplets, predicted by the Dropmix model at a pressure of 1250



psia, result in significantly low. ROCCID results indicate high frequency instabilities are damped when liquid droplets remain in the combustion zone. Empirical droplet size model selected in ROCCID is critical to assessing combustion instability. As shown in Figures 7.a, 8.a, and 8.b, larger RP-1 droplets predicted by the Dropmix model at a pressure of 1250 psia result in stable combustion. However, at these same operating conditions, smaller RP-1 droplets predicted by the Priem model result in unstable combustion. The O/F ratio was discovered to have only slight sensitivity of combustion instabilities. At an operating condition of  $P_c=1250$  psia and  $O/F=2.8$ , the valve assumed for mixing efficiency affects calculated combustion stability (Figure 7.b). In general, as mixing efficiency decreases, combustion instability increases.

Low frequency combustion stability was not affected by propellant supply temperature (Figure 9.a). High frequency stability, however, was affected by propellant supply temperature.

The propellant supply temperature was varied over a range which is expected for a typical ground test facility. Low frequency combustion stability was not affected by propellant supply temperature (Figure 9.a). High frequency stability, however, was affected by propellant supply temperature. The oxidizer, LOX, vaporizes fast and combustion instability was unaffected by supply temperature. However, vaporization rate of the fuel, RP-1, was influenced over the range of propellant

supply temperatures (Figure 9.b). Therefore, calculation of combustion stability was affected.

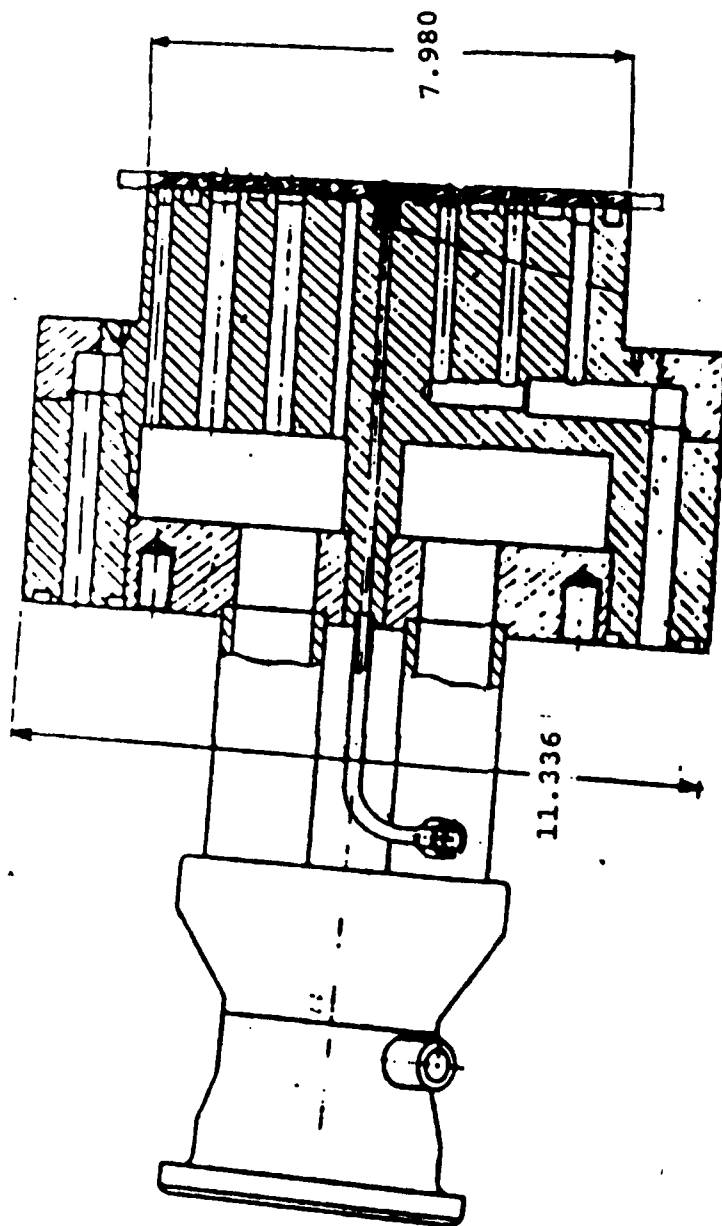
Low frequency results for ROCCID combustion instability analysis are furnished on Figure 10. Mixing efficiencies, O/F ratios, and droplet sizes are compared. As predicted, there was little variance in the minute maximum amplitudes of the low frequency runs.

## 7.0 CONCLUSIONS

The sensitivity of operating parameters on combustion instabilities is a significant issue. The ROCCID computer code provides a low cost route to determining this sensitivity. ROCCID was used to determine the sensitivity of droplet size, mixing efficiency, O/F ratio, propellant temperature, and chamber pressure. The greatest sensitivity to combustion instability was found to be droplet size, propellant supply temperature, and mixing efficiency. The effect of these operating parameters with a different injector type or an added damping device should be explored.

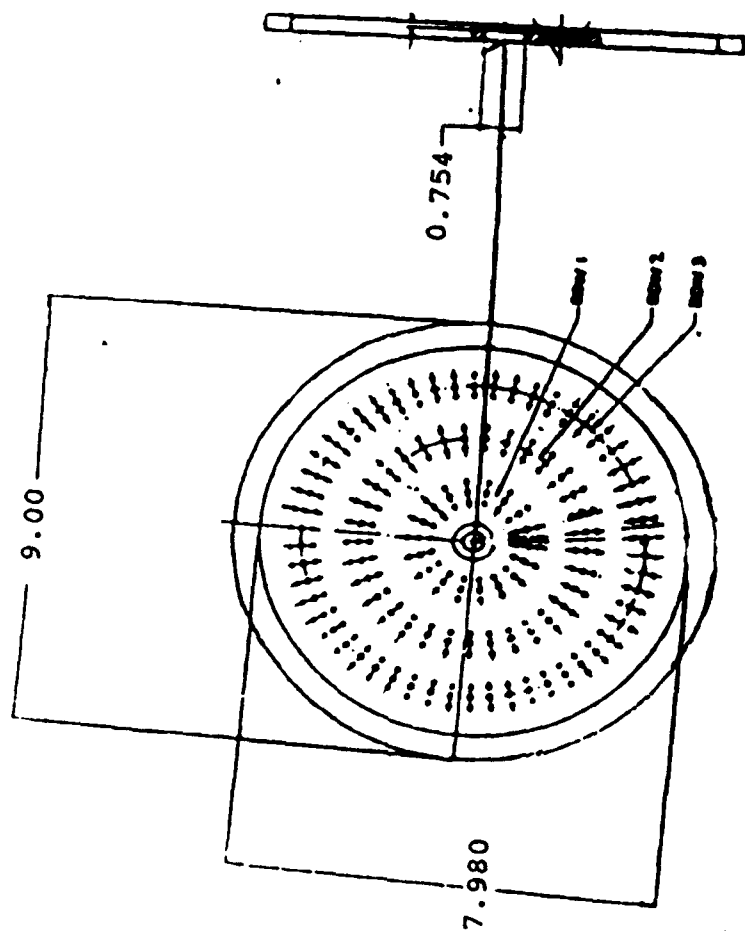
## 8.0 REFERENCES

- 1) Hill, Philip G., and Peterson, Carl R., Mechanics and Thermodynamics of Propulsion; Addison-Wesley, 1965.
- 2) Sutton, George P., and Ross, Donald M., Rocket Propulsion Elements; John Wiley & Sons, 1976.
- 3) Liquid Propellant Rocket Combustion Instability; D.T. Harrje, Ed., 1972.
- 4) Aerojet, LOX/Hydrocarbon Rocket Engine Analytical Design Methodology Development and Validation; August 1990.
- 5) Muss, J.A., Nguyen, T.V., and Johnson, C.W., User's Manual for Rocket Combustor Interactive Design (ROCCID) and Analysis Computer Program; May 1991.



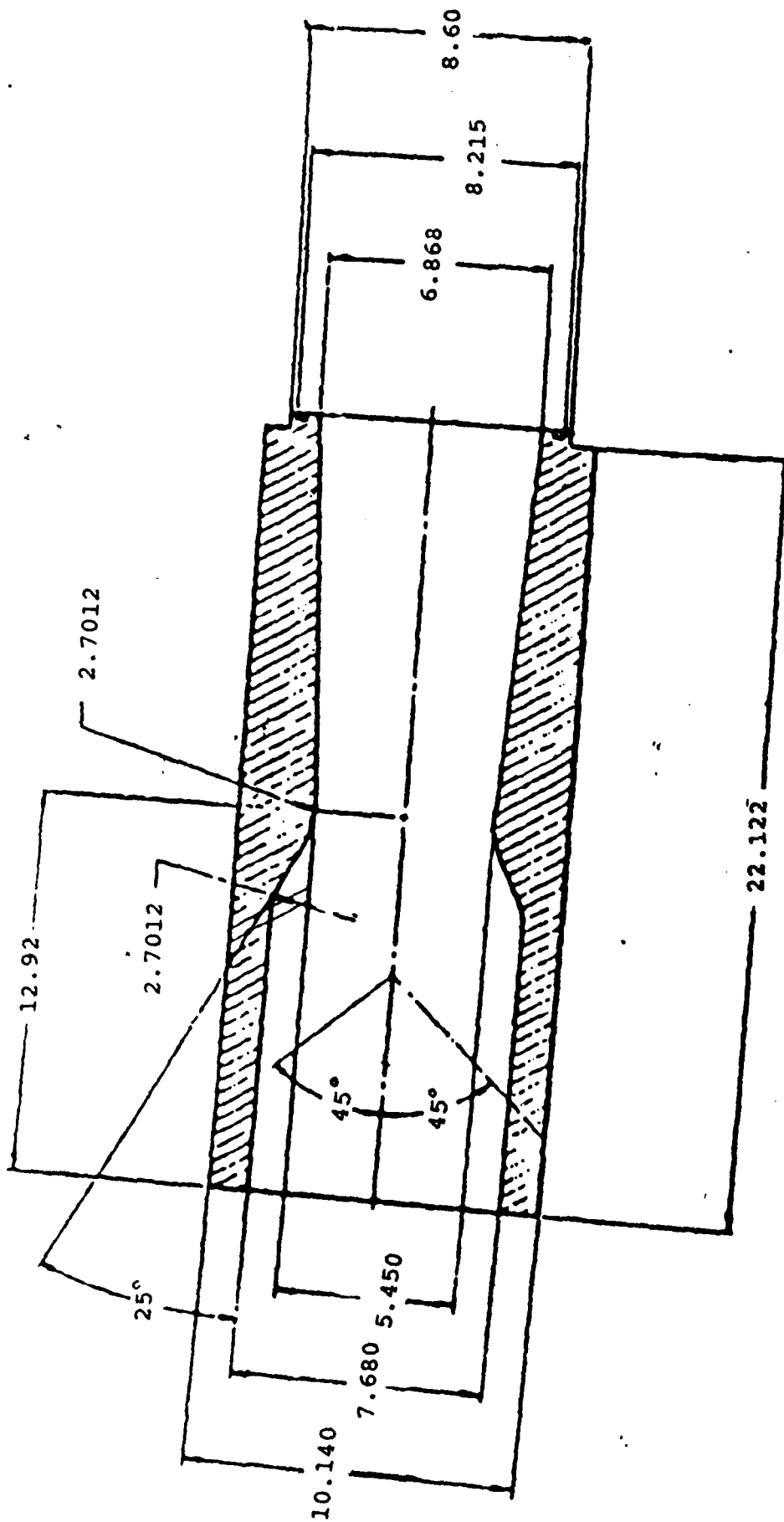
All dimensions are in inches.

Figure 1. Injector Core Assembly



All dimensions are in inches.

Figure 2. Injector Faceplate



All dimensions are in inches.

Figure 3. Combustion Chamber Geometry

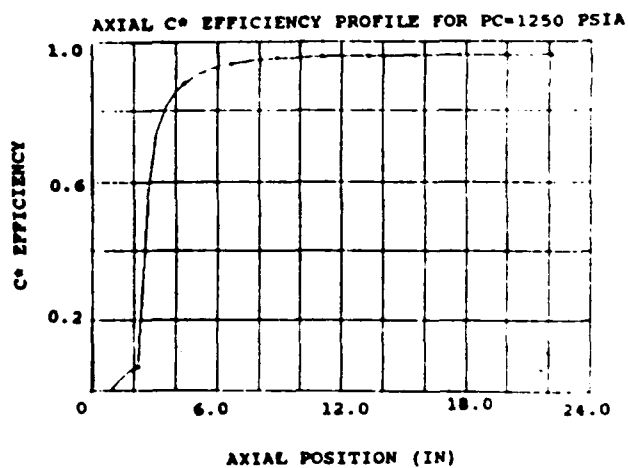
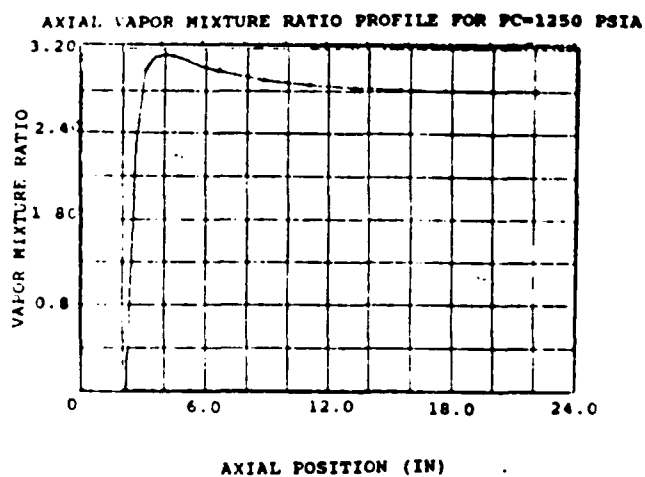
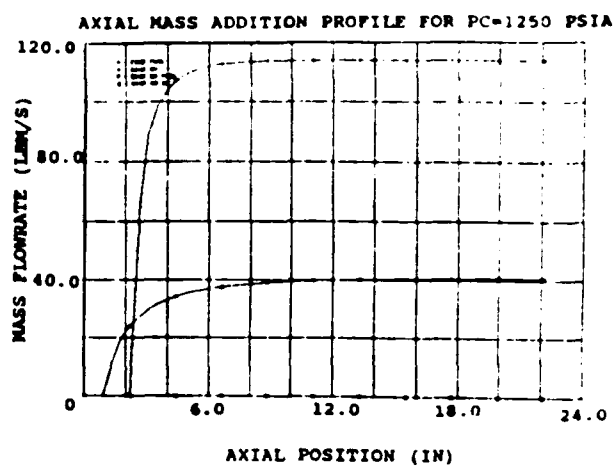
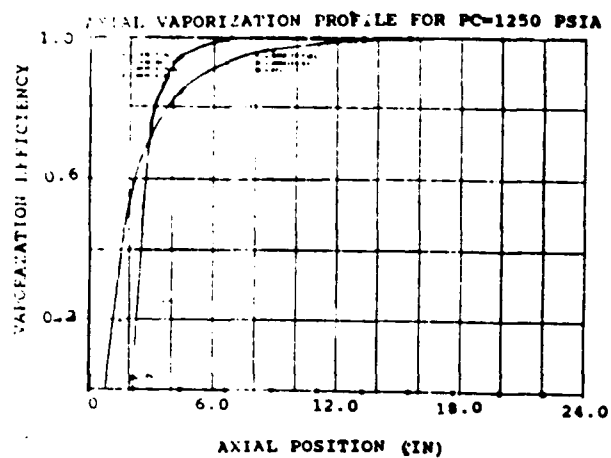


Figure 4. Typical Steady State Combustion Plots

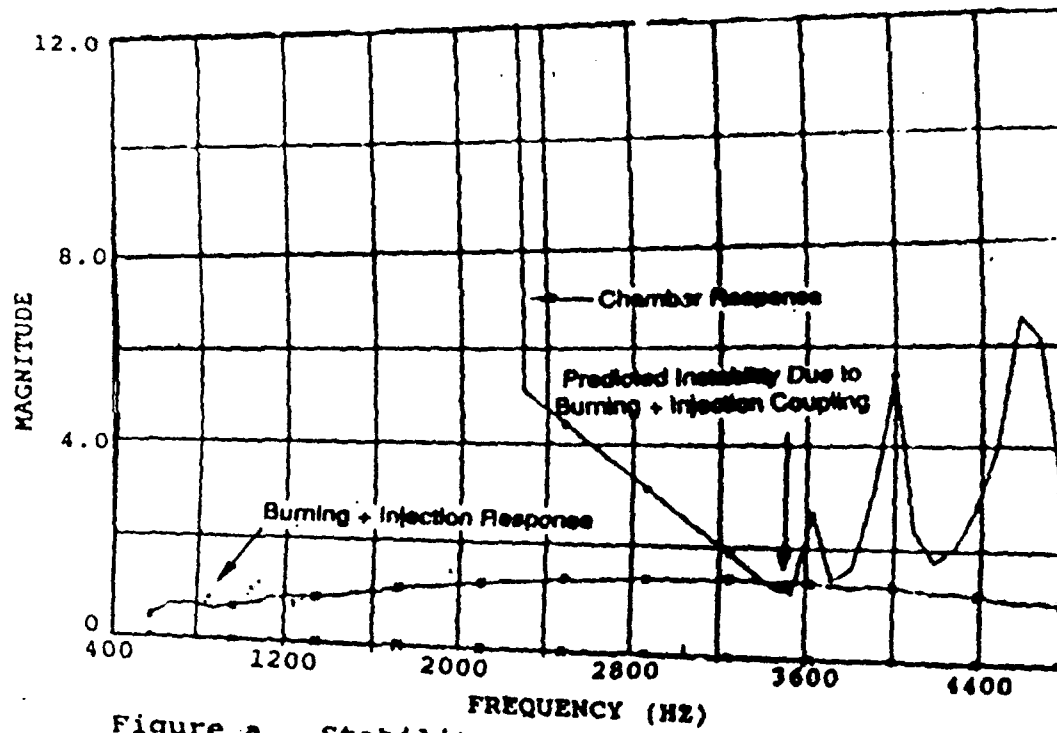


Figure a. Stability Component for PC-750

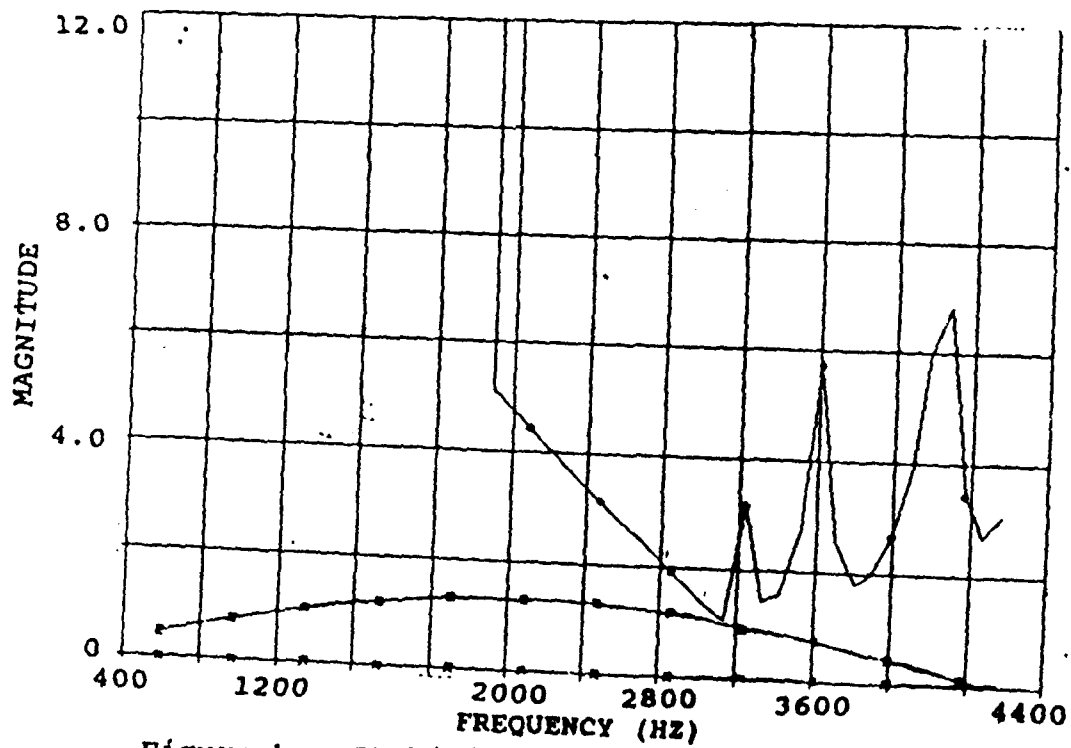


Figure b. Stability Component for PC-1250

Figure 5. 1T + OR Stability Calculation Components



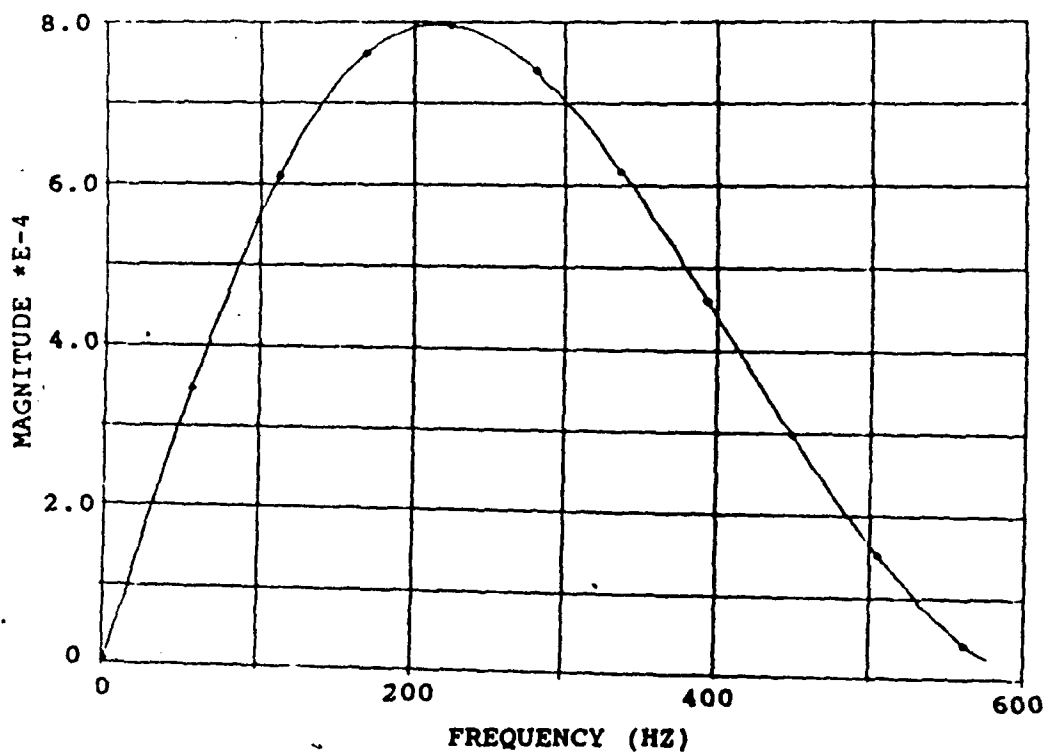
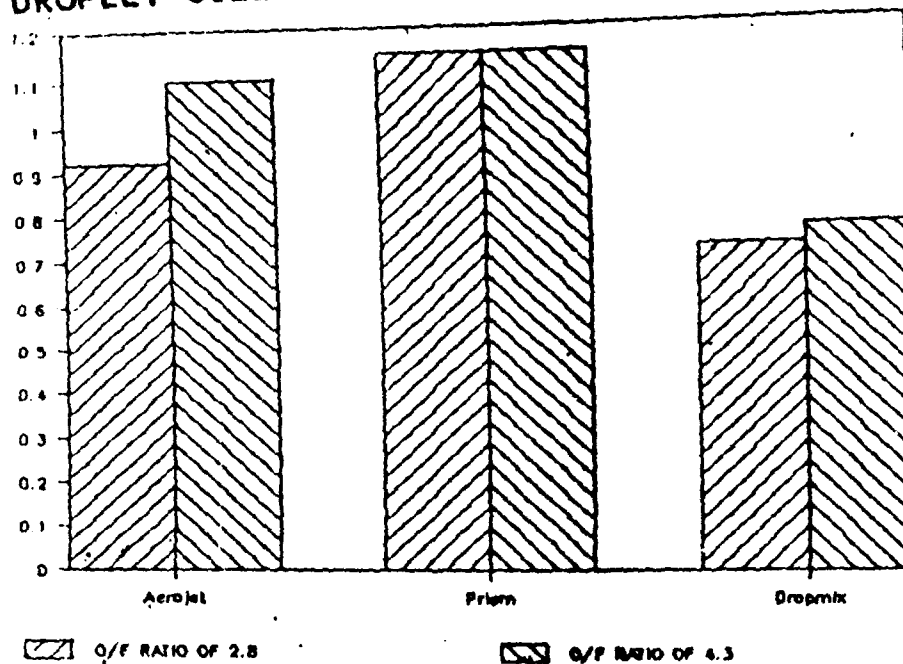


Figure 6. Chug Overall Gain For PC=1250 psia

# **DROPLET SIZE MODEL AND O/F RATIO VARIATION**

Figure 7.a  
MAXIMUM AMPLITUDE



# **MIXING EFFICIENCY AND PRESSURE VARIATION**

Figure 7.b  
MAXIMUM AMPLITUDE

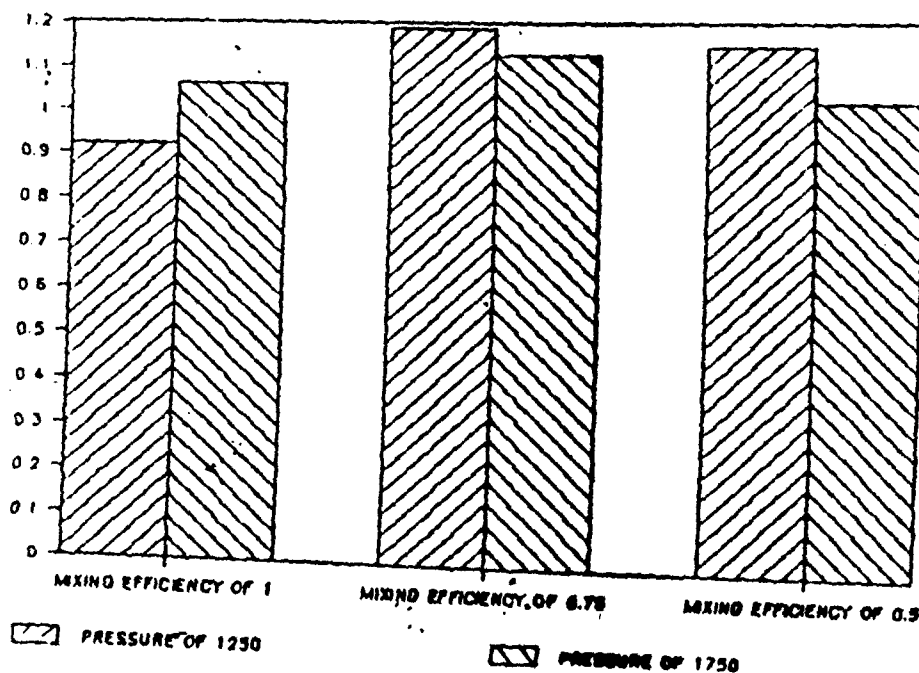
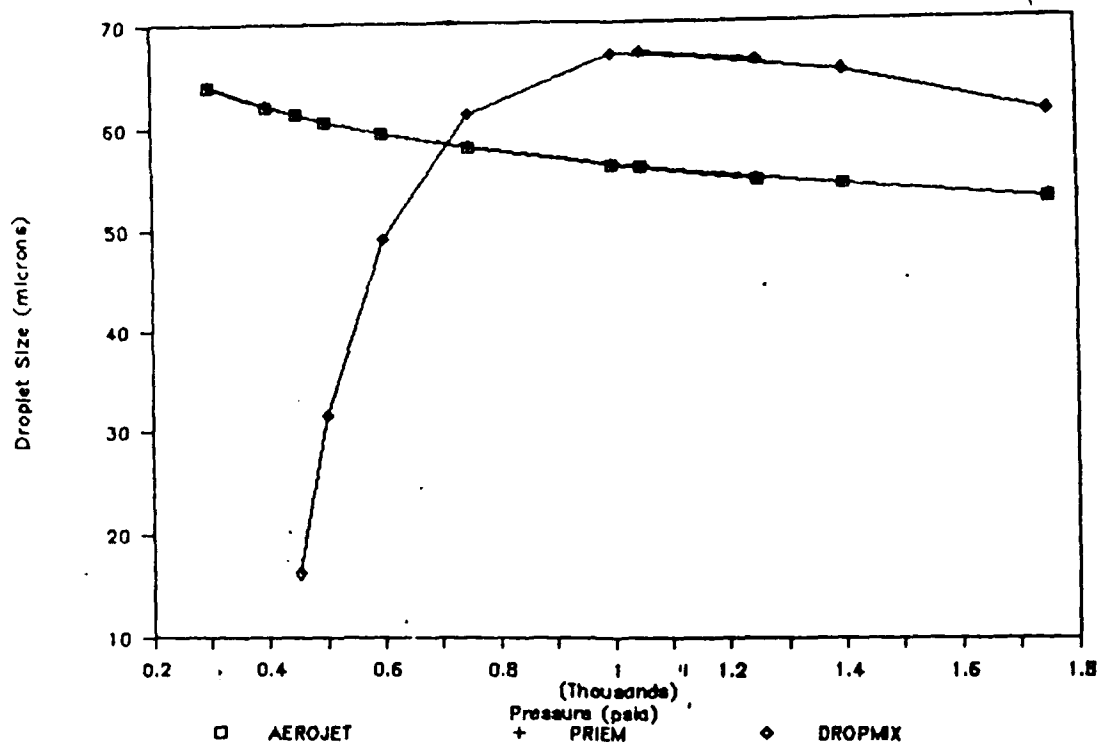


Figure 7. Results of High Frequency Operating Condition Variation

# FUEL DROPLET SIZE AS A FUNCTION OF PRESSURE

Figure a.



# OXIDIZER DROPLET SIZE AS A FUNCTION OF PRESSURE

Figure b.

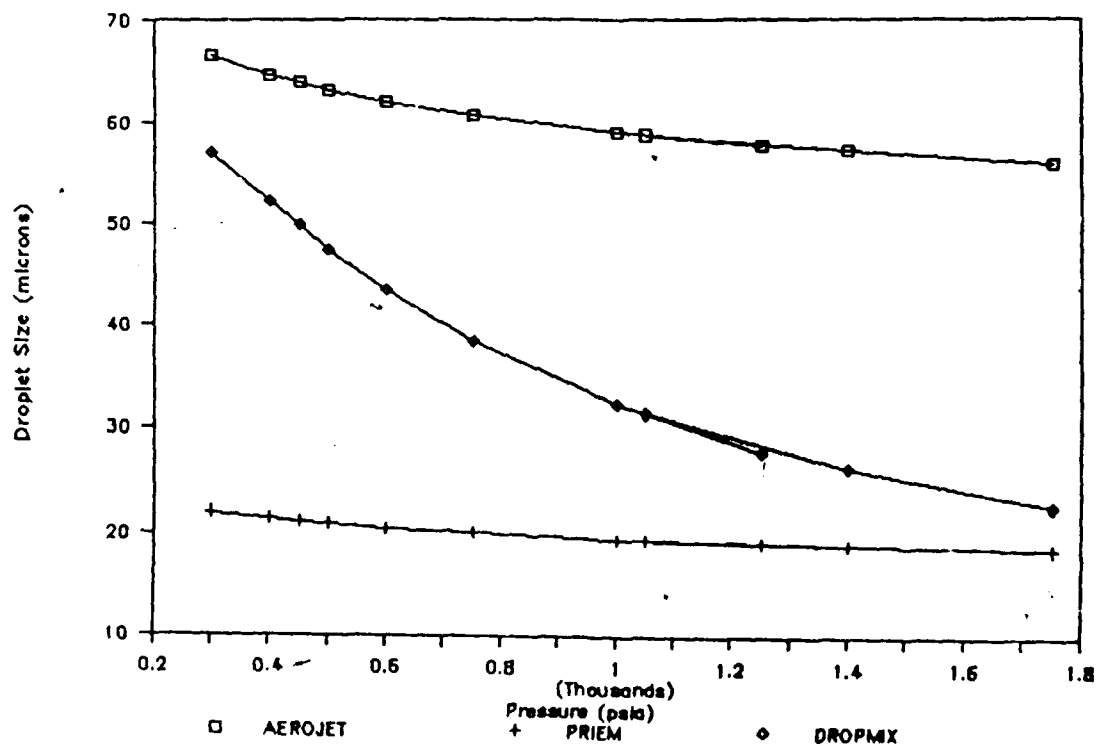


Figure 8. Droplet Size as a Function of Pressure

Figure 9.a  
Maximum Amplitude  $\times 10^{-4}$

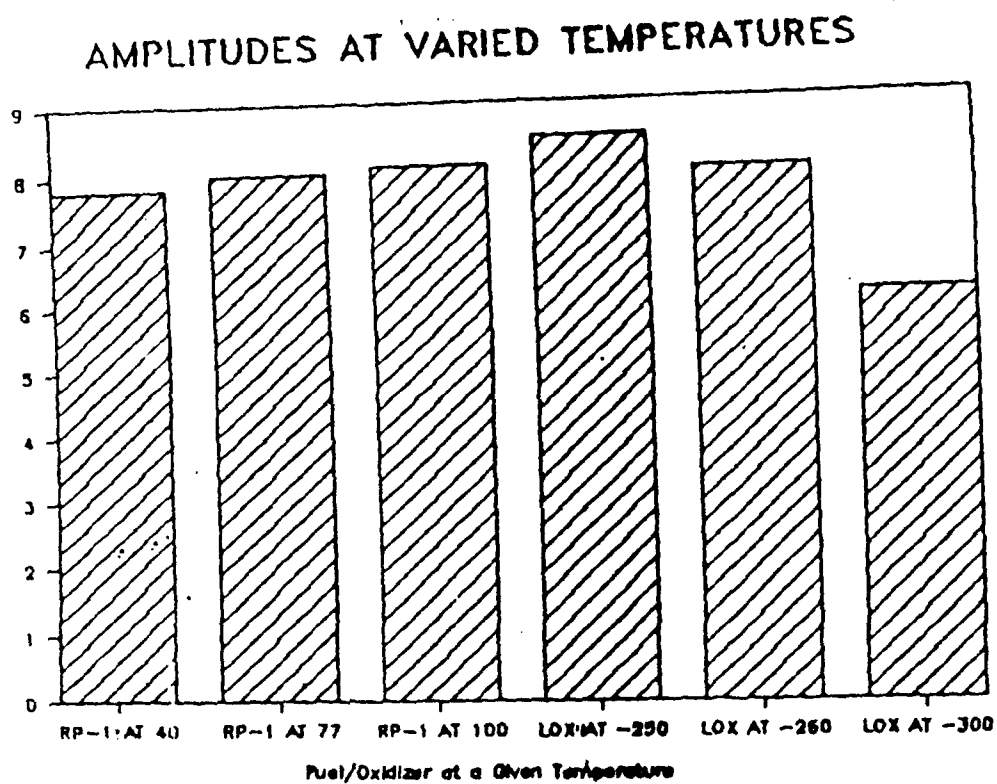


Figure 9.b  
Maximum Amplitude

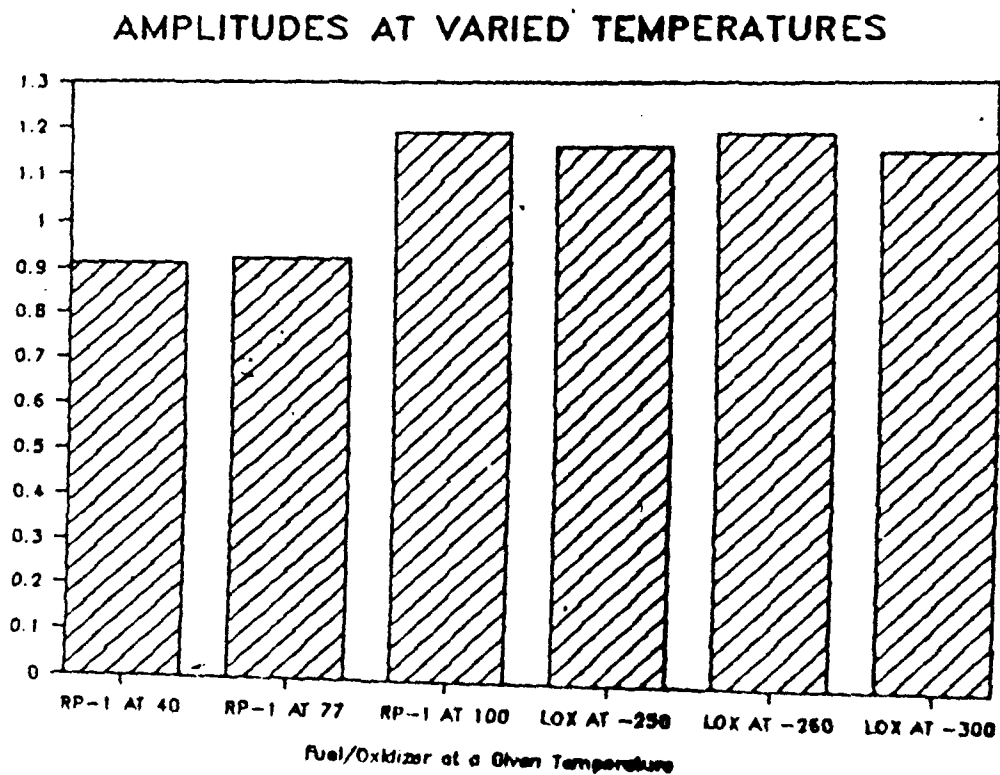


Figure 9. Results of Low and High Frequency Temperature Variation

Figure 10.a

Amplitude  $E^* = 4$

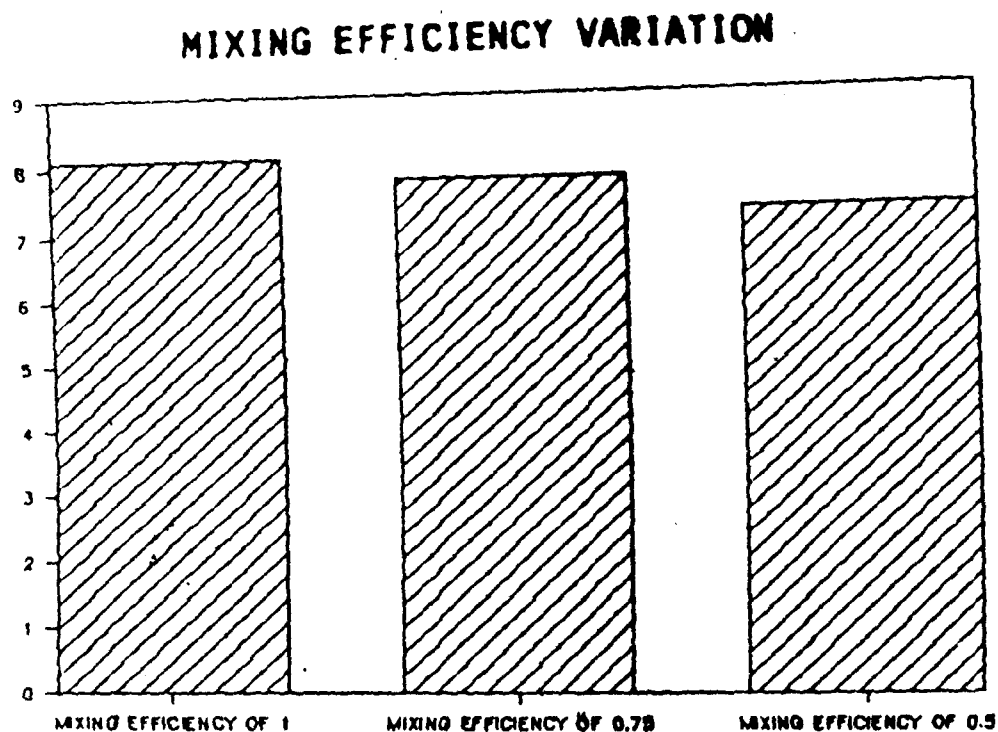


Figure 10.b

MAX AMPLITUDE  $E^* = 4$

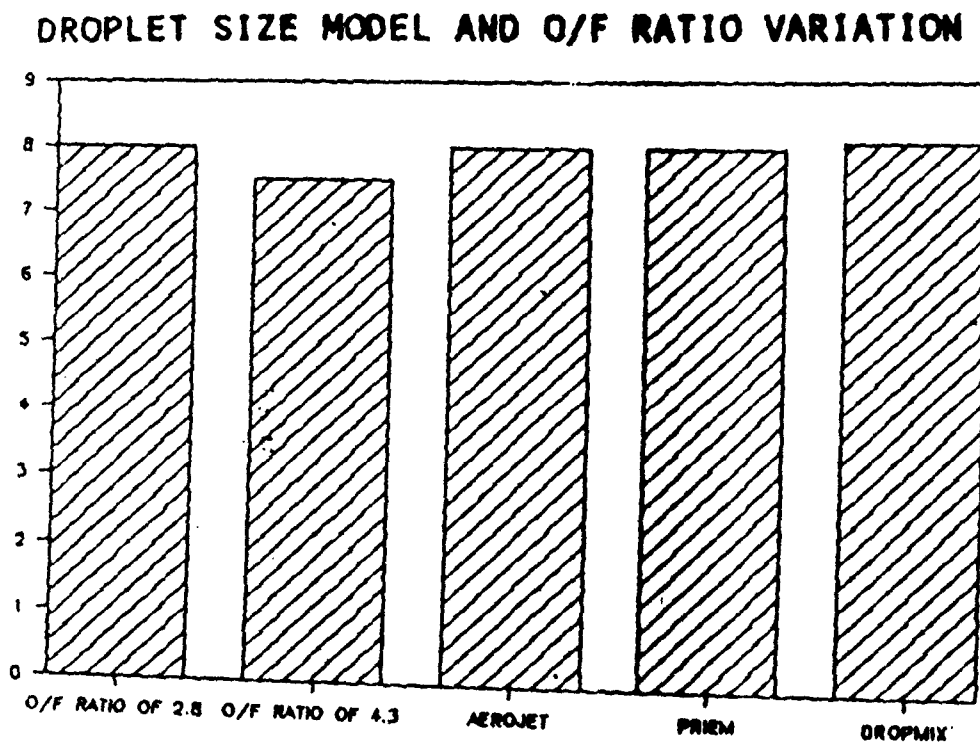


Figure 10. Results of Low Frequency Operating Condition Variation

LOW FREQUENCY COMBUSTION INSTABILITY RESULTS								
Case Number	Chamber Pressure (psia)	Droplet Size	O/F Ratio	Mixing Efficiency	Burning Response Model	Fuel Temp. (F)	Oxidizer Temp. (F)	Max Amp. (*E-4)
1	1250	Aerojet	2.8	1	2-N-TAU	77	-260	8
2	1250	Prien	2.8	1	2-N-TAU	77	-260	8
3	1250	Dropmix	2.8	1	2-N-TAU	77	-260	8.1
4	1250	Aerojet	4.3	1	2-N-TAU	77	-260	7.5
5	1250	Aerojet	2.8	0.75	2-N-TAU	77	-260	7.8
6	1250	Aerojet	2.8	0.5	2-N-TAU	77	-260	7.3

Table 1. Low Frequency Instability Results

HIGH FREQUENCY COMBUSTION INSTABILITY RESULTS								
Case Number	Chamber Pressure (psia)	Droplet Size	O/F Ratio	Mixing Efficiency	Burning Response Model	Fuel Temp. (F)	Oxidizer Temp. (F)	Max Amplitude
1	1250	Aerojet	2.8	1	2-N-TAU	77	-260	0.92
2	1250	Prien	2.8	1	2-N-TAU	77	-260	1.15
3	1250	Dropmix	2.8	1	2-N-TAU	77	-260	0.72
4	1250	Aerojet	4.3	1	2-N-TAU	77	-260	1.1
5	1250	Prien	4.3	1	2-N-TAU	77	-260	1.15
6	1250	Dropmix	4.3	1	2-N-TAU	77	-260	0.76
7	1250	Aerojet	2.8	0.75	2-N-TAU	77	-260	1.19
8	1250	Aerojet	2.8	0.5	2-N-TAU	77	-260	1.15
9	1250	Aerojet	2.8	1	CRP	77	-260	0.82
10	1250	Aerojet	2.8	1	2-N-TAU	40	-260	0.91
11	1250	Aerojet	2.8	1	2-N-TAU	100	-260	1.19
12	1250	Aerojet	2.8	1	2-N-TAU	77	-240	1.19

Table 2. High Frequency Instability Results

TUNNEL 4T DATA TABULATION AND GRAPHS

JONATHAN ROBERT SANDERS

MENTOR: WARREN GILBERT

ARNOLD ENGINEERING DEVELOPMENT CENTER

08-09-91



My job was to write a software program which could access Tunnel 4T event data and convert this data into a usable spreadsheet format. The final part of the job was to produce tabulated and graphical results. The Tunnel 4T event data consisted of such parameters as Mach number, start-ups, shutdowns, air on test condition time and energy, and valve positions from twenty-six different tests over the last year. Each test was run individually, and the spreadsheet formatted data was then downloaded onto a Zenith 248 personal computer for presentational purposes. The Tunnel 4T event data will be used to establish operational norms and indicate potential improvement areas.

My mentor, Warren Gilbert, was a very good and competent one. He was extremely intelligent, courteous, and cautious of what I was doing. We continuously discussed what I was doing in order to ensure that I knew what I was doing and that I felt comfortable with my work. He treated me with considerable respect. He also appeared to be very confident with me, and he treated me in a professional manner. I would like to sincerely thank him.

When my mentor was unavailable to help me, Dan Cunningham volunteered his efforts generously. He was nice to me. I would also like to thank him.

## TUNNEL 4T DATA TABULATION AND GRAPHS

SUMMER APPRENTICE JONATHAN ROBERT SANDERS

Arnold Engineering Development Center has three Propulsion Wind Tunnels 4T, 16T, and 16S. The number relates the size of the test section in feet in each individual tunnel, while the letter reveals a speed, T- transonic or S- supersonic. The tunnels are capable of simulating almost every condition that an aircraft might encounter while in flight. Models are placed inside the tunnels, and tests are run to determine the most efficient and reliable structure of the to-be-made aircraft. This on-ground testing reduces the required flight testing of a new aircraft and, therefore, the development costs.

Over the past year, much need has come about for the availability of a working software program that could access Tunnel 4T event data and produce readily available tabulations and graphs. This was my job. Twenty-six different tests have been individually run in Tunnel 4T in the last eighteen months, and the event data from these tests has done little more than sit on a disk due to the lack of having a software program and its capabilities.

Before my work efforts this summer, operational personnel had limited available data to determine causes of various operational problems; but with the implementation of the new software program, plant operational data is now readily available in the form of spreadsheets, tabulations, and graphs; thus, the engineer's job in ascertaining problems has been made much easier. Examples of tabulated

data and graphed parameters range from test condition parameters to valve positions in the wind tunnels.

One specific quantity that is graphed is energy distribution. In this graph the energy use associated with start-up, change condition, on condition, off condition, and shutdown are converted into dollars. The ideal situation when a test is run is to be on test conditions nearly one hundred percent of the time and to minimize change condition and off condition time. If change condition and off condition time and energy are continuously high, tunnel time, energy, and, thus, money can be wasted. This type of data reveals to engineers areas that need improvement.

When testing requires more time and energy because of problems in the control algorithms, engineers can look at the data and at the graphs to determine how to keep such parameters as Mach number and pressure within desired tolerances. If any parameter deviates, time and energy have been wasted. If test conditions stay on line (no parameter deviation), engineers can finish a given test quicker. This means that the work is much more efficiently done, thus resulting in saved dollars not only to corporations who hired AEDC to do its testing, but also to individual taxpayers like yourselves.

The first job that I was given was to write a software program which could read in data that had been recorded during tests. The idea of having an effective means to easily access data from particular tests was becoming more popular and important. If any information was needed from any previous test, test engineers had to go to great pains just to get the information that they needed.

The program performed three functions, reading data from particular events and tests into a spreadsheet, reading the spreadsheets' data into summary spreadsheets, and reading the summary spreadsheets into one grand summary spreadsheet which was to then be downloaded onto a personal computer so that graphs could be made. The data would be taken from a file which had been made and placed onto a storage disk during a test, run through the software program, and placed into several spreadsheets.

In order to write this software program, I used a previously written Tunnel 16T software program as a foundation and incorporated numerous changes, such as changed cell references, changed formats, changed programming, insertion of an extra data event into the program, and changed printing commands. I completed this portion of my task and continued.

The next portion of my job was to run the various tests through the software program. This was done by simply entering a spreadsheet, typing in a command, and invoking the new software program. Many tests took several hours to run due to the enormous amount of data and due to the length that the test ran in the tunnel. After the tests were through running, many parts of each spreadsheet had to be hand checked with a calculator in order to ensure that the data was being read in correctly. After this each test was tabulated. When the spreadsheet was to my mentor's and my satisfaction, the next portion of my job began.

I then downloaded the grand summary spreadsheet onto a personal computer. The spreadsheet was of such immense size that this was necessary. The terminal that I had run all of the tests on had very

few graphics capabilities. We needed graphs to better see how properly the tunnels were working. The personal computer had a very good graphics system but could not handle the immense size of the spreadsheet in the spreadsheet's present condition, thus, indicating why we did not start on the personal computer.

The download process consisted of three parts and was carried out by a macro, a chain of commands performed one after the other automatically by the computer rather than by the user. The first part dealt with parsing each individual test. When the parse method was complete, the second downloading step became effective. Here the parsed data for each individual test was incorporated into a single spreadsheet, and summations by Mach number were made. These summaries were downloaded into a final summary spreadsheet for presenting the data.

When the download process finished, the graphs were almost ready to be plotted; but first cell references had to be changed in order that the plot equations plotted the correct ranges of data. When the time came for the graphs to be drawn, a printing command was invoked on the personal computer; and they were drawn and printed. In addition, tabulations of the summarized data were made.

In conclusion, I finished all of my assigned work. I completed writing the software program; I ran the twenty-six tests through the software program; I downloaded the data onto a personal computer; I adjusted the plots; I graphed the ranges; and I tabulated the data. The uses of the tabulated event data and of the graphs are multifarious and will be of tremendous help when put into real situations.

I learned many valuable lessons this summer. One is that each individual part of a software program is extremely important. For instance, the software program would not run all of the way through due to some problem in it. I had left one "recalculate" symbol out, and the program aborted. I also learned that engineers have paperwork galore; and, thus, organization is a must. The work is quite demanding but not impossible. I learned some Fortran language. I have learned a considerable amount about how to use spreadsheets. I have also gained Lotus skills which I know will be useful. I had a great time working this summer.

EDT Editor. RSX-11M-Plus, Volume 3B. Digital Equipment Corporation, September 1983.

Lotus 1-2-3 Reference. Release 2.2. Lotus Development Corporation, 1989.

Saturn-Calc. Saturn Systems, Inc., August 1988.

TITLE: COMPUTER AIDED DYNAMIC DATA MONITORING AND ANALYSIS SYSTEM

1991 USAF-RDL SUMMER HIGH SCHOOL APPRENTICESHIP  
PROGRAM FINAL REPORT

Prepared by: Jason M. Scott  
Mentor: Lt. Greg Nordstrom  
Research Location: AEDC - DOTP

ABSTRACT

My work this summer in the High School Apprenticeship Program was centered around the CADDMAS Project. The project is planned to be used for engine testing at AEDC. The CADDMAS system will have the ability to process all data from the various sensors in an engine test. My focus this summer was the user interface and graphics software of this system. The user interface is important because it is one of the most visible parts of the system. There are many features which must be incorporated and yet it should stay easy to use. The graphics software should produce displays with many features but stay easy to comprehend.



## 1. INTRODUCTION:

Turbine Engine testing generates massive volumes of data at very high rates. A single engine may have several thousand sensors and test periods lasting for several hours. Due to the huge amount of data this produces, it is not feasible to digitally store this information. The Computer Assisted Dynamic Data Monitoring and Analysis System (CADDMAS) project will process the data on-line at the speed of acquisition by the use high-computation algorithms. These algorithms effectively compress the data into useful information that is easier to store.

The CADDMAS project is made up of several main parts:

1. *The Numeric Processing Element (NPE)* uses a *T800* transputer and a *Zoran Signal Processor* [10] to supply the most of the computational power of the system.
2. *The Graphics Processing Element (GPE)* incorporates a *Texas Instruments TMS34010* [9] with a dual frame buffer. This processing element performs quick, smooth display of dynamic signals.
3. *The Storage Processing Element (SPE)* uses a *SCSI* interface to store large quantities of information.
4. *The I/O Processing Element (IOPE)* provides a variety of configurable digital parallel I/O lines.
5. *The Front End Processor (FEP)* performs the A/D conversion, and uses *Motorola 56001 Digital Signal Processors* [6] to complete the high-rate peak detection, and time and frequency domain alarm-level monitoring.

My main emphasis involved the development of the user interface and software for the Graphics Processing Element. The GPE provides a high performance, high

# GPE Block Diagram

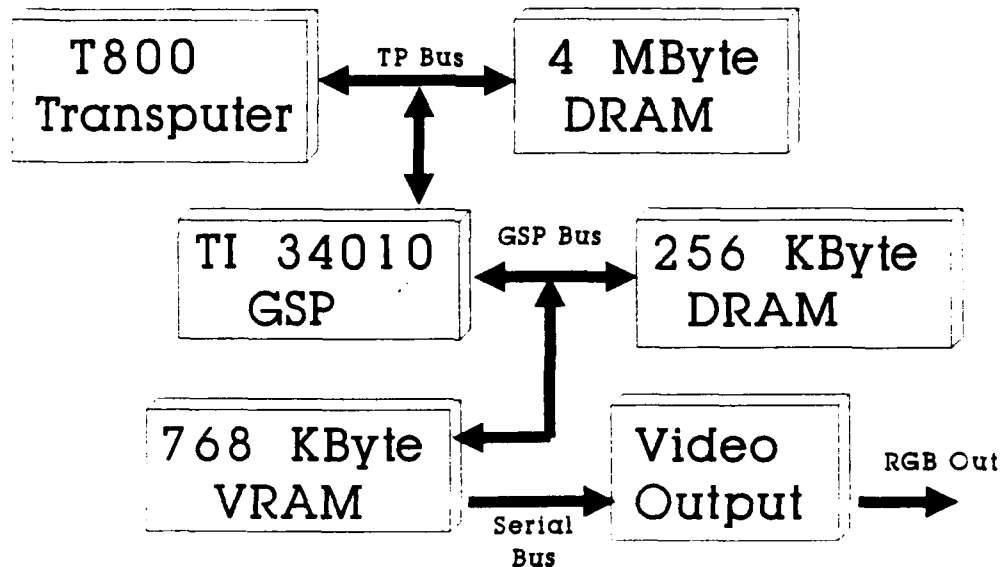


Figure 1: Graphics Processing Element Block Diagram

resolution, color display interface. This processor receives processed information from many sensors and integrates the data into a display screen. System users need high resolution graphics to display data from the large number of sensors. The engineers need high-speed rendering to display dynamically changing information so that it will appear as continuous time to the operator. This device also integrates information from several clusters of Numeric Processing Elements.

The GPE is a dual-CPU computer, with high-speed general purpose computation, and high-speed, high resolution color graphics. The graphics resolution is 1024x768 with 16 simultaneous colors. The GSP has 256K Bytes of DRAM for program and data storage functions. There is enough video memory (768K bytes) for dual frame buffers. With the double buffering technique, all drawing can occur in the invisible screen. Only completely drawn images are visible on the display. This reduces the flicker caused by repeated clearing and redrawing of the graphics screen. (See figure 1).

Also on the GPE is a T800 transputer that passes buffers that contain display list commands and data to be plotted to the GSP. These commands are low-level operations such as point, line, box, and text operations. There are functions for line, bar, and waterfall plots. There is software supporting non-overlapping, tiled windows. Drawing operations are clipped outside the currently selected window.

I started this summer by refining and adding many new features to the user interface. The user interface was already in place but needed several minor modifications to make it easier to use. It was written in the C language using the CXL Extensions that provide pull-down menus, windows, and many other graphical interface functions. The interface is used to control the graphical output of the system and act as a shell between the user and GNUPLOT.

The interface controls such functions as managing up to sixteen windows on each of the two graphics displays and placing up to five of any of the available plots in any of these windows. It controls the placement of titles, labels, and special labels for campbell diagrams. It can save the configurations of window positions, labels, ect. into a file for easy recall. The user interface controls the digital storage of CADDMAS with various functions such as play, record, pause, ect. This sends the appropriate message to the Storage Processing Element which in turn controls an external hard disk.

The user interface has windows positioned on its screen to simulate the windows on the graphics displays. I added some new features to control the size and number of windows on both display screens. These different window configurations are now saved as text files so that it is very easy to add new configurations or edit old ones using any text editor.

I also have added a capture function to the interface that allows the user to save a specified number of blocks of data to a DOS file. This data can be from any channel at a specified IRIG time and can skip a number of blocks at regular intervals to include a wider range of information in less space. Later this plot can be viewed using a stand-alone version of GNUPLOT on any PC or workstation or

may be used as a data file in a spreadsheet program.

Here are some of the other modifications made:

- the capability to display up to five plots in one window simultaneously
- the option of having a grid on each individual window
- a new menu item 'Clear' to clear all parameters on the active screen
- the ability to set the active window on the user interface by simply clicking on it with the mouse, and a second click brings up that window's parameters menu.
- a new menu item 'Command' that allows the user to bypass the user interface and send commands directly to GNUPLOT.

Later in the summer I began work on modifying GNUPLOT, a plotting package from the Free Software Foundation which has been ported to the GPE. The GNU software runs on the T800, with a special TI34010 graphics chip device interface to translate the display list commands. The GNUPLOT program is fully portable to almost any computer by simply adding that computer's drawing routines. GNUPLOT can be used to display multiple plots on workstations or personal computers during a test. After a test, the engineer can take a copy of a certain section of the test data he is interested in back to his personal computer and view it using GNUPLOT.

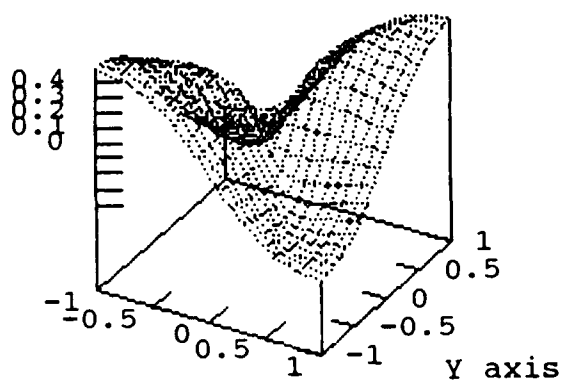
Last year, I modified the GNUPLOT program to do several specific functions that AEDC test engineers need to have available to them. In particular, subroutines to plot Campbell and phase Campbell diagrams were added. These are plots with frequency on the y-axis, rotor speed on the x-axis, and stress on the z-axis. The z-axis is represented as a vertical line with varying amplitudes on the graph. In a phase Campbell there is phase data instead of stress data. In a phase Campbell the phase data is represented with an arrow tip pointed in the relative direction of the phase.

This summer I have worked on adapting GNUPLOT to support 3-D graphics for the display of Campbell Diagrams and other multidimensional data. I have also worked on making my modifications for campbell diagrams easier to put into future versions of GNUPLOT. The latest version of GNUPLOT now also supports the XWINDOWS environment on the workstation. I also modified GNUPLOT to display multiple, user defined, plots on the PC. I did this by adding several new commands for defining the size and position of the area to be plotted in. These multiple plots are also compatable with HPGL and other printing formats so they can be sent directly to a printer. (See figure 2).

Other things I did include helping prepare for demonstrations of the CADDMAS system this summer. Two briefings were presented. The first briefing was presented to the AEDC potential users of the system. The second briefing audience was AEDC upper-level management and outside visitors from General Electric Evansville Division. Along with this effort, we made a short, 5 minute, videotape to introduce the capabilities of the CADDMAS system to a wider range of people. I also went with several others on a trip to Vanderbilt University for a demonstration of the Multigraph Kernal (MGK) which is being considered for use in the CADDMAS system. This environment for execution of parallel programs was developed by Vanderbilt researchers to confront the difficulties of parallel programming.

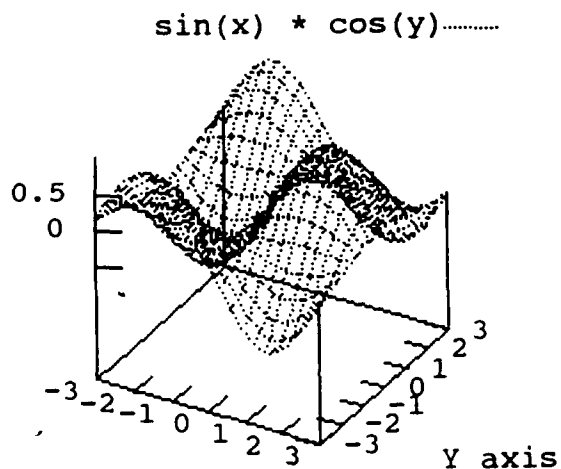
3D gnu plot demo

3D gnu plot demo



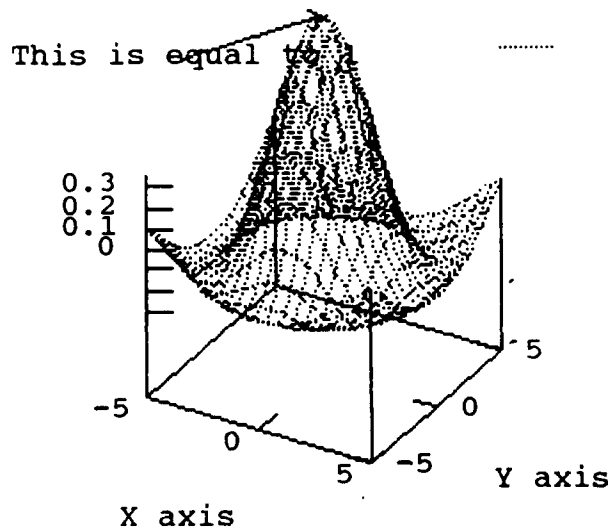
X axis

Sinc function

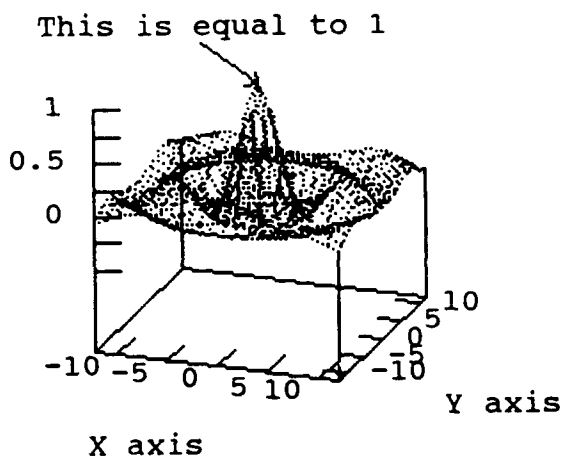


X axis

Sinc function



X axis



X axis

Figure 2: GNUPLOT - multiple plots on PC screen

### Acknowledgements

I wish to thank the Air Force Systems Command and the Air Force Office of Scientific Research for sponsorship of this research. Research and Development Labs must also be commended for their concern and help to us in all administrative and directional aspects of this program. My experience was rewarding and enriching because of many different influences. Lt. Greg Nordstrom provided me with support, encouragement, and a truly enjoyable working atmosphere. I would also like to thank Ben Abbott and Ted Bapty for teaching me many new things and for making this summer very enjoyable. Finally, I would like to thank Major Hart for all his help in insuring that my experience at AEDC would be memorable.

- [1] Biegl, C.: "Design and Implementation of an Execution Environment for Knowledge-Based Systems" Ph.D. thesis, Department of Electrical Engineering, Vanderbilt University, Nashville, TN., Dec., 1988.
- [2] Abbott, B., Biegl, C., Sztipanovits, J.: "Multigraph for the Transputer".  
Proceedings of the Third Conference of The North American Transputer Users Group.  
Santa Clara, Calif., April 1990.
- [3] Abbott, B., Bapty, T.: "High-Speed Parallel Signal Processing",  
FINAL REPORT, 1990 USAF-UES SUMMER FACULTY RESEARCH PROGRAM.  
Conducted by the Universal Energy Systems, Inc. Contract No: F49620-88-C-0053. Aug., 8, 1990.
- [4] Bailey, D., Barton, J.: "The NAS Kernel Benchmark Program", National Aeronautics and Space Administration, Technical Memorandum. Aug. 1985.
- [5] Nordstrom, G. AEDC/DOTR:  
*Personal communication (as well as various notes and schematics).*  
July, 1991.
- [6] INMOS Limited: DSP56000 Digital Signal Processor User's Manual, 1989.
- [7] INMOS Limited: Transputer Reference Manual, 1989.
- [8] Karsai, G.: "Hierarchical Description Language (HDL) User's Manual" Dept. of Electrical Engineering, Vanderbilt University, Technical Report #87-004, 1987.
- [9] Texas Instruments Incorporated: **TMS34010 User's Guide.**
- [10] Zoran Corporation: **ZR34345 32-Bit IEEE Floating-Point Vector Signal Processor User's Manual.**



# **HIGH PERFORMANCE LIQUID CHROMATOGRAPHY**

**JENNIFER M. TRENT**

**MENTOR: PENNY MILLER**

**AEDC**

**9 AUG 91**

## INTRODUCTION

Chromatography is the process by which the components of a sample are carried by a mobile phase, which may be either liquid or gas, through a stationary phase, either solid or liquid. The components are then separated based on their relative attractions to each of the phases. In column chromatography, the stationary phase is held in a long narrow tube, or column, through which the mobile phase is forced by pressure or gravity. Column chromatography may be divided into two groups, liquid and gas, which refer to the physical state of the mobile phase. An advantage to using liquid column chromatography over gas column chromatography is that samples which are not volatile (able to become a gas) and samples that are thermally unstable may be run. High performance liquid chromatography (HPLC) is a form of liquid chromatography in which the mobile phase is forced through the stationary phase in the column under a very high pressure. The result is a much higher speed, resolution, and sensitivity in the analysis; therefore, HPLC is becoming a much more desirable method of analysis.

The invention of chromatography dates back to the turn of the century when a Russian botanist, Mikhail Tswett, used the technique to separate various plant pigments, such as chlorophylls and xanthophylls, by passing solutions of these compounds through a gas column packed tightly with calcium carbonate. He observed many colored zones indicating a separation of the pigments. Although the applications of chromatography in general have grown tremendously in the last four decades, liquid column chromatography has had a very slow growth due to quicker, more sensitive chromatographic methods. HPLC has corrected the speed and sensitivity problems, however, and is rapidly becoming a more utilized analytical technique.

HPLC is an extremely useful tool in preparing samples for analysis or for performing certain analyses itself; therefore, a liquid chromatograph (the machine used in HPLC) would be very important in a chemical lab. However, the one at the AEDC Chem Lab had been inoperative for about ten years. My initial assignment was to revive this old machine, to get it cleaned out and working. After that, I was to learn the theory and principles behind HPLC, learn the functions of the liquid chromatograph and its many parts,

understand and experiment with the applications of HPLC and how they relate to real-life analyses, and finally, be able to interpret the results of my experiments based on what I had learned. I found that there are many ways to save time and effort using HPLC.

## **DISCUSSION**

### **PROCESS AND APPARATUS**

The machine used to perform HPLC is called a liquid chromatograph. Refer to Appendix A. During the chromatographic process, solvent (or mobile phase) is constantly running through it. The solvent is propelled by the solvent delivery system (or pump). The injector is used to inject sample into the system. Then the solvent sweeps the sample into the column, which is packed tightly with the stationary phase. Each component in the sample is attracted to the stationary phase in different ways, so the components move through the column at different rates, thus separating them. As each component leaves the column, it passes through a detector which then sends a signal to the recorder. The recorder then draws a peak for each component. This is called a chromatogram. It is a plot of time versus detector output. In the meantime, the combination of the solvent and the sample, called the eluate, leaves the detector and may be collected.

During my research I dealt with three types of detectors. The first, a fluorometer, measures the amount of fluorescence in a material. It could not be used for all samples since not all fluoresce. The second one I used was a spectrophotometer, which measures the absorbance of ultraviolet light transmitted through the sample. I also used a differential refractometer, which measures the refractive index of a sample.

The primary focus of my work with the liquid chromatograph dealt with the use of three columns and the applications of each. The first column I used was a reverse-phase column. In reverse-phase, the mobile phase is polar while the stationary phase is non-polar. Normally it is the other way around. The

second column I used was a gel-permeation column. In gel-permeation, the stationary phase is held in the small spaces of a polymeric solid. This column is used to separate a sample by molecular weight, the higher molecular weight appearing on the chromatogram first. The last column I used was a prep-scale column, which is basically the same as a reverse-phase column, but it has a much larger capacity. Using a prep-scale column, a contaminated sample, such as a fuel, may be run, and after the contaminants pass through the detector, the pure sample may be collected for further testing. Other columns would not hold a large enough sample for more testing.

## **ANALYZING A CHROMATOGRAM**

Refer to Appendix C. A chromatogram is a plot of time versus detector output. A peak is recorded as the detector senses a component of a sample. RT represents the retention time, or the time from the point of injection to the peak maximum. Taking into consideration the flowrate of the solvent, one can also find the retention volume of the material. Under very controlled conditions, the retention time or volume can be used to identify the components, since the two are characteristic of a given compound.

The area of each peak is also listed. Since the area of a peak is also proportional to the component's concentration, one can determine the exact concentrations of the components. This may be done by graphing or by formula.

## **COLUMN EXPERIMENTS**

This section will introduce some of the major experiments I ran involving the three aforementioned types of columns. Only the conditions of the experiments will be discussed. Results are found in a later section.

### **REVERSE-PHASE COLUMN:**

#### **Solvents/Flowrates**

Refer to Appendix B. During my research, I learned that some samples required more than one solvent for a good separation. This is a sample of naphthalene, benzoic acid, and diphenylamine mixed in ethanol. In the first

chromatogram, only acetonitrile is used as the solvent. In the second chromatogram, 50% acetonitrile and 50% water are used as the solvent

#### Anthracene Standards

Refer to Appendix C. These are two concentrations of anthracene standards. The first is 10ppm and the second is 50ppm. A 100 ppm was also made. My mentor then made some unknown concentrations of the anthracene and I was to determine their concentrations.

#### Crane Engine Oil

Refer to Appendix D. The chem lab was brought a sample of HDO-30 oil collected from a crane engine. It was believed to be contaminated with diesel fuel. The task was to confirm that there was diesel fuel and then to determine exactly how much. The first chromatogram is pure diesel fuel. The second is pure clean HDO-30 oil. The third is the contaminated sample brought to the lab. All three were dissolved in methylene chloride. the first two samples run were to allow me to see the peaks and RT's of pure compounds, and the last sample run was for determining if diesel was present. Diesel standards of 250, 350, 400, and 500ppm concentrations were used to determine the concentration in the sample.

#### **GEL-PERMEATION COLUMN:**

##### Polypropylene Filter

Refer to Appendix E. This is a chromatogram of a length of polypropylene filter dissolved in hot fuel. I also ran samples of polypropylene dissolved in fuel, polypropylene dissolved in toluene, and polystyrenes dissolved in carbon disulfide.

#### **PREP-SCALE COLUMN:**

##### Sample Clean-up

Refer to Appendix F. This is a sample of naphthalene and benzidine dissolved in ethanol. The test may be run on any number of types of samples.

## RESULTS

### REVERSE PHASE COLUMN:

#### Solvents/Flowrates

Appendix B. In the first chromatogram, one can distinguish three separate components, but there is hardly any separation. In the second one, there is a significant difference. The addition of water to the mobile phase yielded a much better separation. After I tried 100% acetonitrile, I tried 90% acetonitrile/10% water, then 80% acetonitrile/20% water, etc. The more water I added, the better the separation. However, it can be noticed that the 50% acetonitrile/water sample has a much longer retention time; therefore, increasing the resolution decreased the speed. One obstacle to working with HPLC is determining the conditions which will yield the best efficiency.

#### Anthracene Standards

Appendix C. The 10ppm and 50ppm standards look very much alike. The tiny peak represents acetone, used to dissolve the anthracene. The large peak represents anthracene itself. Because the 50ppm standard was more concentrated, the peak is larger, five times larger to be exact. Its area is 224,448,633 while the 10 ppm's area is 53,319,101. The variation is fairly direct. Using these two standards and a 100ppm standard, I was able to find the concentrations of my unknowns in two ways. I did this first by graphing. I made a graph of concentration vs. area, and it was linear. Next I ran the unknowns and plotted their areas. The corresponding concentrations were the concentrations of the unknowns. Then I determined the concentrations by formula. The RF, or response factor, of a sample is found by dividing the area by the concentration. I did this for the 10, 50, 100 ppm standards and then average them to find the average RF. Then I divided the area of each unknown by the average RF to find their concentrations. Both procedures were fairly accurate. This procedure can be helpful in finding the concentration of a contaminant in a sample. The next examples typifies this.

#### Crane Engine Oil

Appendix D. The first peak shown on all three chromatograms represents the methylene chloride (see similar RTs). By examining the first chromatogram, one can see that the diesel is composed of several peaks with

RT's from 4.07 to 5.31. Now look at the second chromatogram. During the same amount of time, no peaks appeared. A peak for the oil would have shown up later, probably much later. This just showed that the diesel and oil did not look the same. Now look at the third chromatogram. As you can see, the peaks for the diesel are present in the oil. Then in order to determine the concentration of the diesel, four diesel standards were made: 250, 350, 400 & 500 ppm. The areas of the three distinct middle peaks were used to find RF's. Using a formula, I found that the concentration of the diesel fuel in the oil was about 9.2%. If the previous method had been used to determine the presence of diesel, we would have had to consider the specific gravities, solubility in water, flash point, refractive Index, and viscosity just to determine its presence. The HPLC is obviously much less time consuming.

#### **GEL-PERMEATION COLUMN:**

##### Polypropylene Filter

Appendix E. The GPC column is used to separate by molecular weight. In this sample, I dissolved a length of polypropylene filter in hot fuel. The first peak is the one that is shown, and it occurred after about 21 minutes. On a chromatogram, the higher the molecular weight, the sooner the peak appears. For this reason, the peak shown is representative of the polypropylene filter, which has a higher molecular weight than the fuel. If I had let the sample run longer, eventually the fuel would have appeared. A real-life application of this experiment is the fact that in the past, there have been several problems with polypropylene filter being dissolved in fuel storage facilities. If the liquid chromatograph had been working, the fuel could have very easily been analyzed.

#### **PREP-SCALE COLUMN:**

##### Sample Clean-up

Appendix F. Although it appears that there are three peaks, there are actually only two, the first two, which represent the naphthalene and benzidine. The small "peak" at the end is just an impurity. It didn't even register as a peak on the chart at the bottom. Because I was running the sample through a prep-scale column, which has a much larger capacity, I was able to use a larger sample - 25 ul instead of the usual 5 or 1ul. Because so much is used, the sample can be "cleaned". For instance, say we have a

sample of fuel which is contaminated, and we need the pure fuel to run a certain test. A large sample may be sent through the liquid chromatograph, and as soon as the chromatogram stabilizes, the impurities will have left the detector. The sample, clean from impurities, may then be collected as it leaves the detector, and other tests may be performed on it. Along with the observance of contaminants in samples, the liquid chromatograph will most likely be used for the purpose of sample clean-up.

## CONCLUSION

During the past eight weeks that I have spent working at Arnold Engineering Development Center under the RDL High School Apprenticeship Program, a large number of things have been accomplished. First, the liquid chromatograph is now able to be used for analyses. A precious resource was being wasted as the machine sat for ten years collecting dust, but now it may be used again. I learned the theory and principles behind HPLC through manuals and instruction. Through experimentation, I learned of numerous applications of HPLC and how the machine could increase efficiency at the Chem Lab. Some applications include determining the concentration of a component of a sample, recognizing contaminants through observing chromatograms, separation by molecular weight, and sample clean-up for further analyses. These would all be very helpful in increasing productivity and efficiency in the Chem Lab.

Probable future use of the liquid chromatograph in the AEDC Chem Lab will mostly revolve around the distinguishing of the presence and concentration of contaminants in samples sent for analysis and cleaning samples with the prep-scale column for further testing.

The personal benefits of this program are tremendous. I have learned several things about science, chemistry, engineering and their real-world applications. It is very beneficial to see hands-on how our backgrounds in science may eventually be applied to a career. Foremost, however, is the sense of accomplishment in knowing that I really contributed to the operations of AEDC.

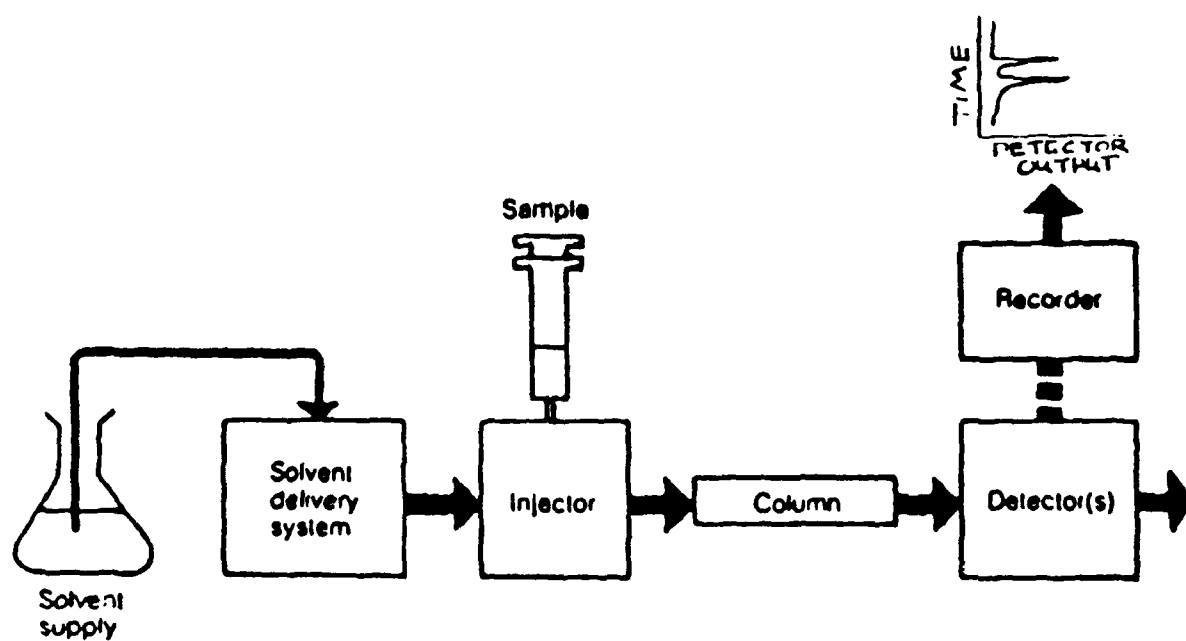


## BIBLIOGRAPHY

1. Brann, Robert D. Introduction to Instrumental Analysis. New York: McGraw-Hill Book Company, 1987.
2. Hadden, Nina, et al. Basic Liquid Chromatography, New York: Varian Aerograph, 1971.
3. Skoog, Douglas A. Principles of Instrumental Analysis, 3rd ed. New York: Saunders College Publishing, 1985.

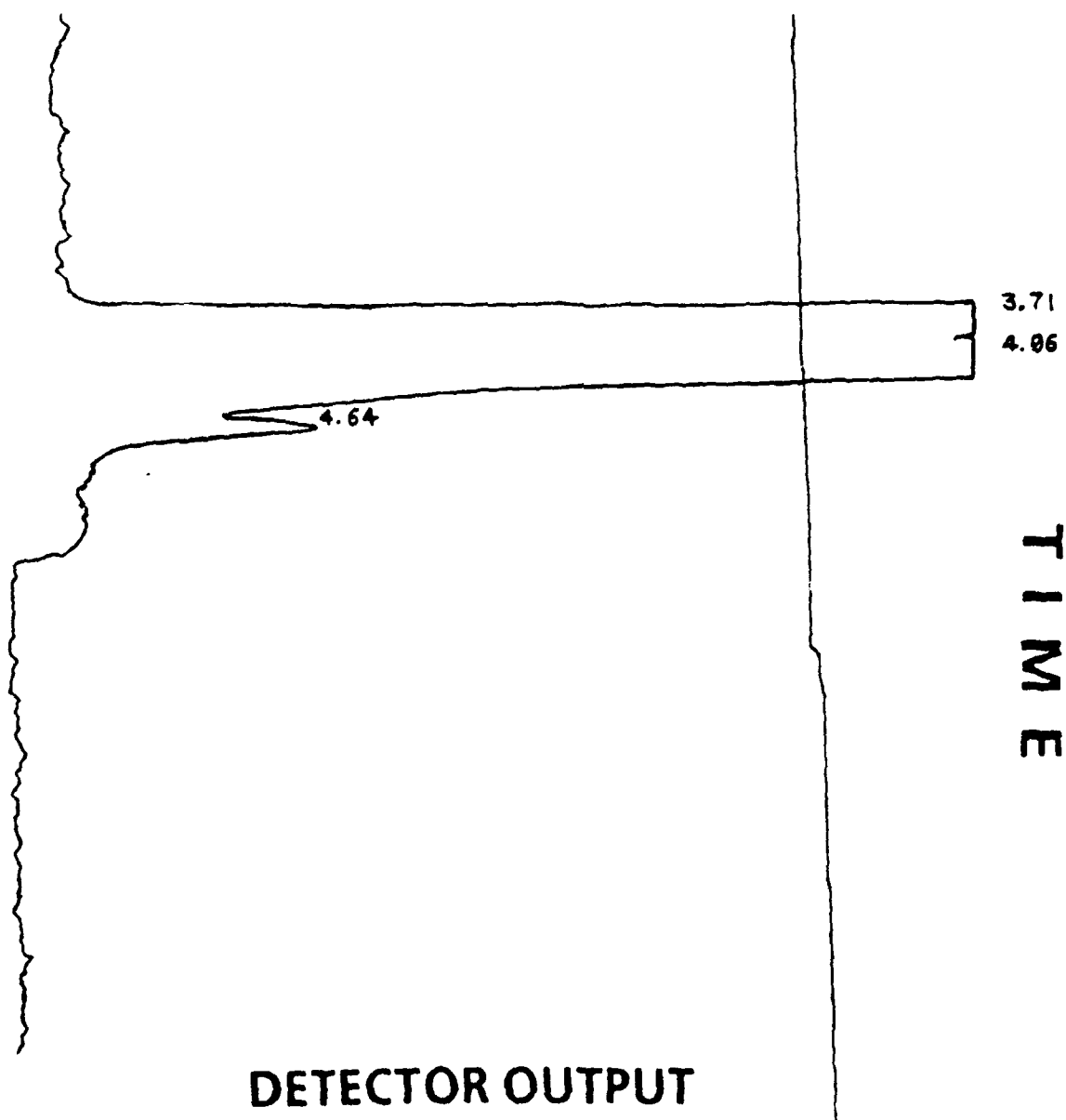
# **APPENDIX A**

# LIQUID CHROMATOGRAPH



## **APPENDIX B**

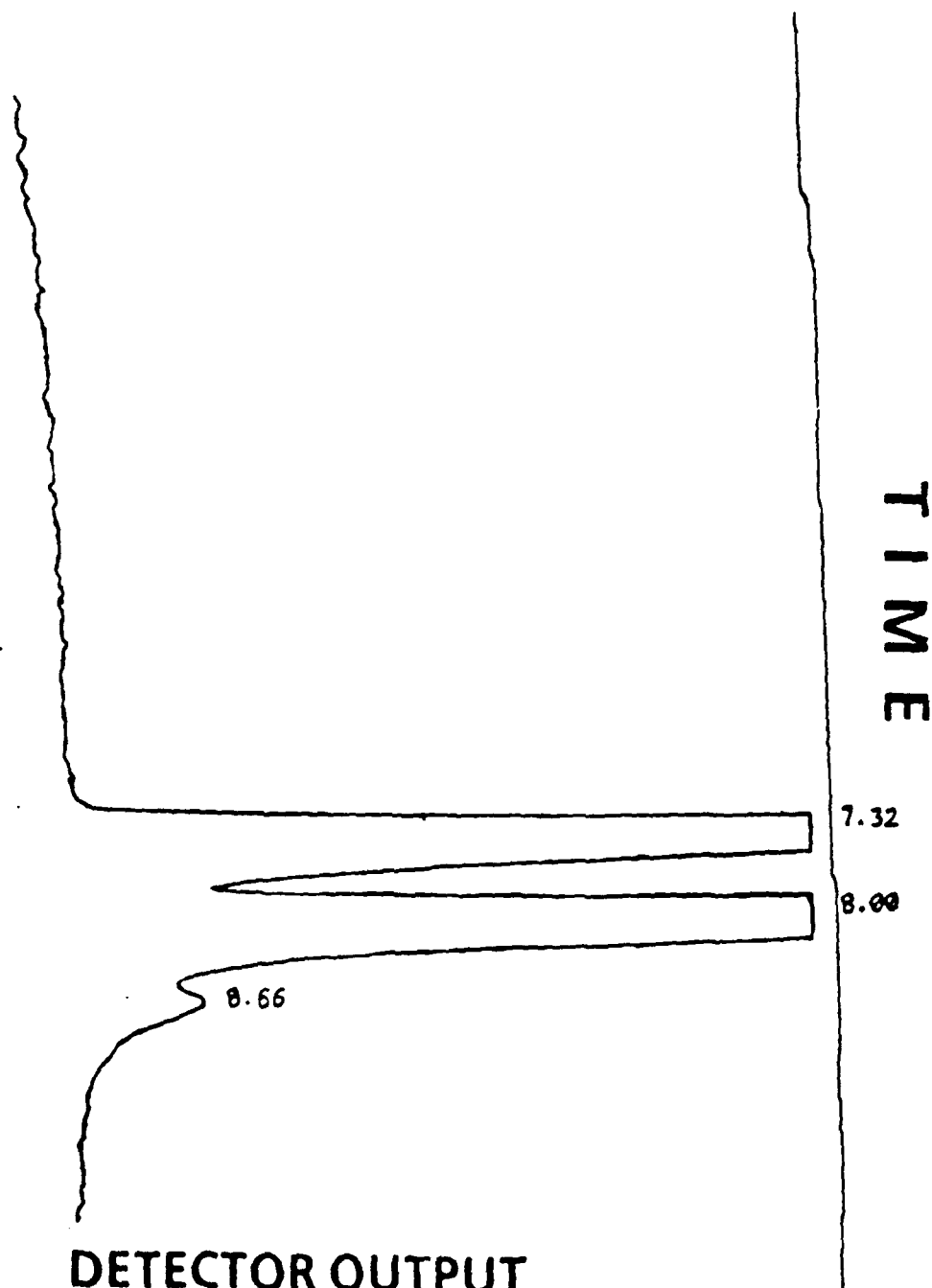
# Naphthalene, Benzoic Acid, Diphenylamine in Ethanol; 100% Acetonitrile Solvent



## EXTERNAL STANDARD QUANTITATION

PEAK#	AMOUNT	RT	EXP RT	AREA	RF
	11041.40000	3.71		11041537 F	0.000000E0
	13209.20000	4.06		13209314 F	0.000000E0
	276.19200	4.64		276192 L	0.000000E0
TOTAL	24526.80000				

# 50% Acetonitrile, 50% Water



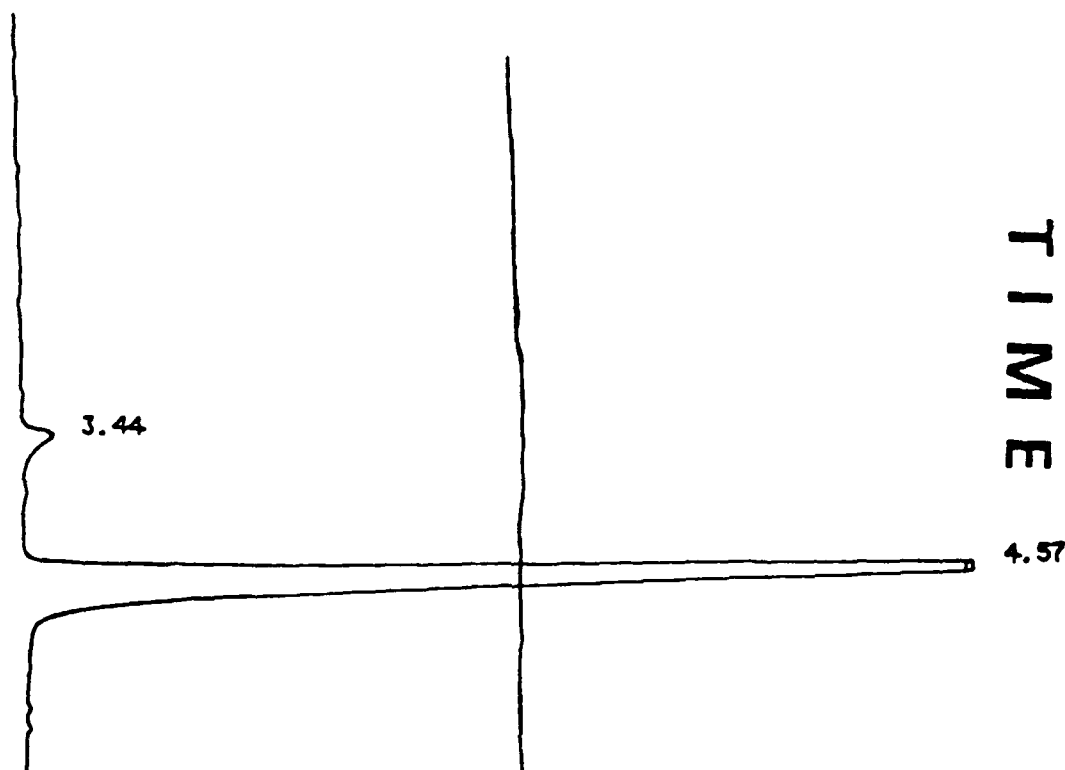
## INTERNAL STANDARD QUANTIFICATION

AREA	AMOUNT	TIME	CONC	TIME	CONC
1115.31000	1.00	7.32	1.000000E0	7.32	1.000000E0
1115.31000	1.00	8.00	1.000000E0	8.00	1.000000E0
1115.31000	1.00	8.66	1.000000E0	8.66	1.000000E0
1115.31000	1.00	8.66	1.000000E0	8.66	1.000000E0

Copy available to DTIC does not  
 permit fully legible reproduction

## **APPENDIX C**

# Anthracene, 10ppm.



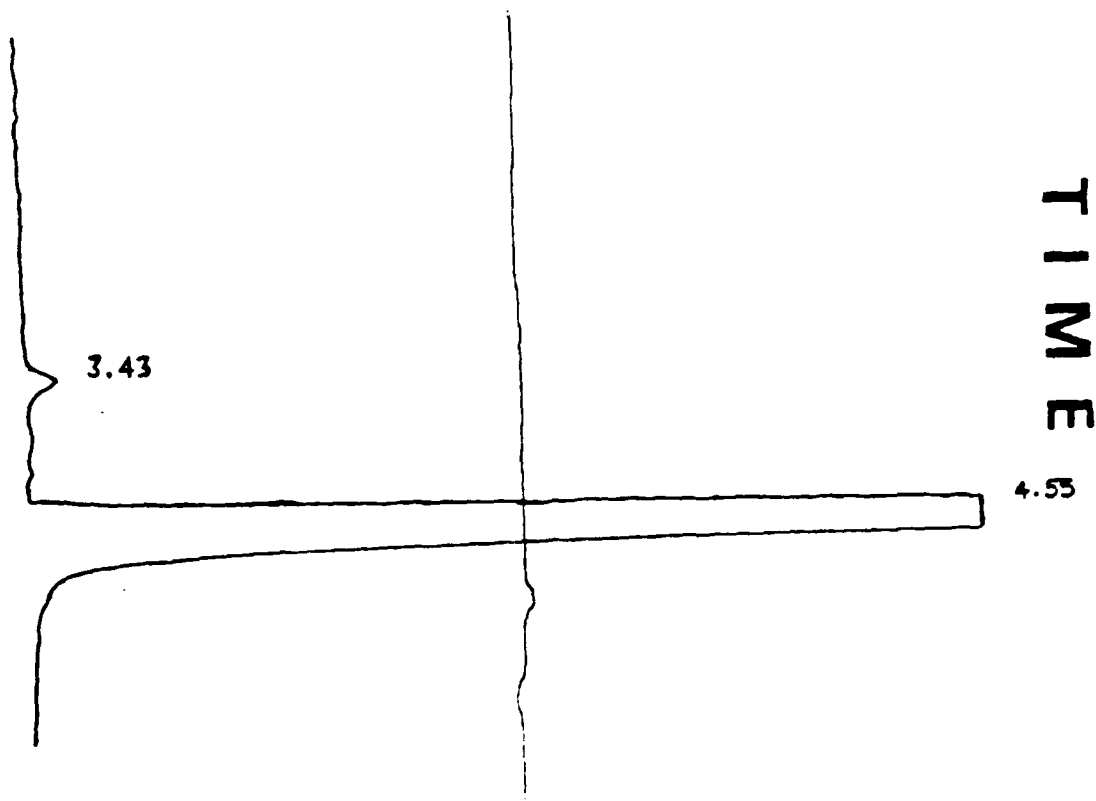
## DETECTOR OUTPUT

### EXTERNAL STANDARD QUANTITATION

PEAK#	AMOUNT	RT	EXP RT	AREA	RF
	1471.91000	3.44		1471911 L	0.000000E0
	53318.70000	4.57		53319101 L	0.000000E0
TOTAL	54790.60000				



# Anthracene, 50 ppm.



## DETECTOR OUTPUT

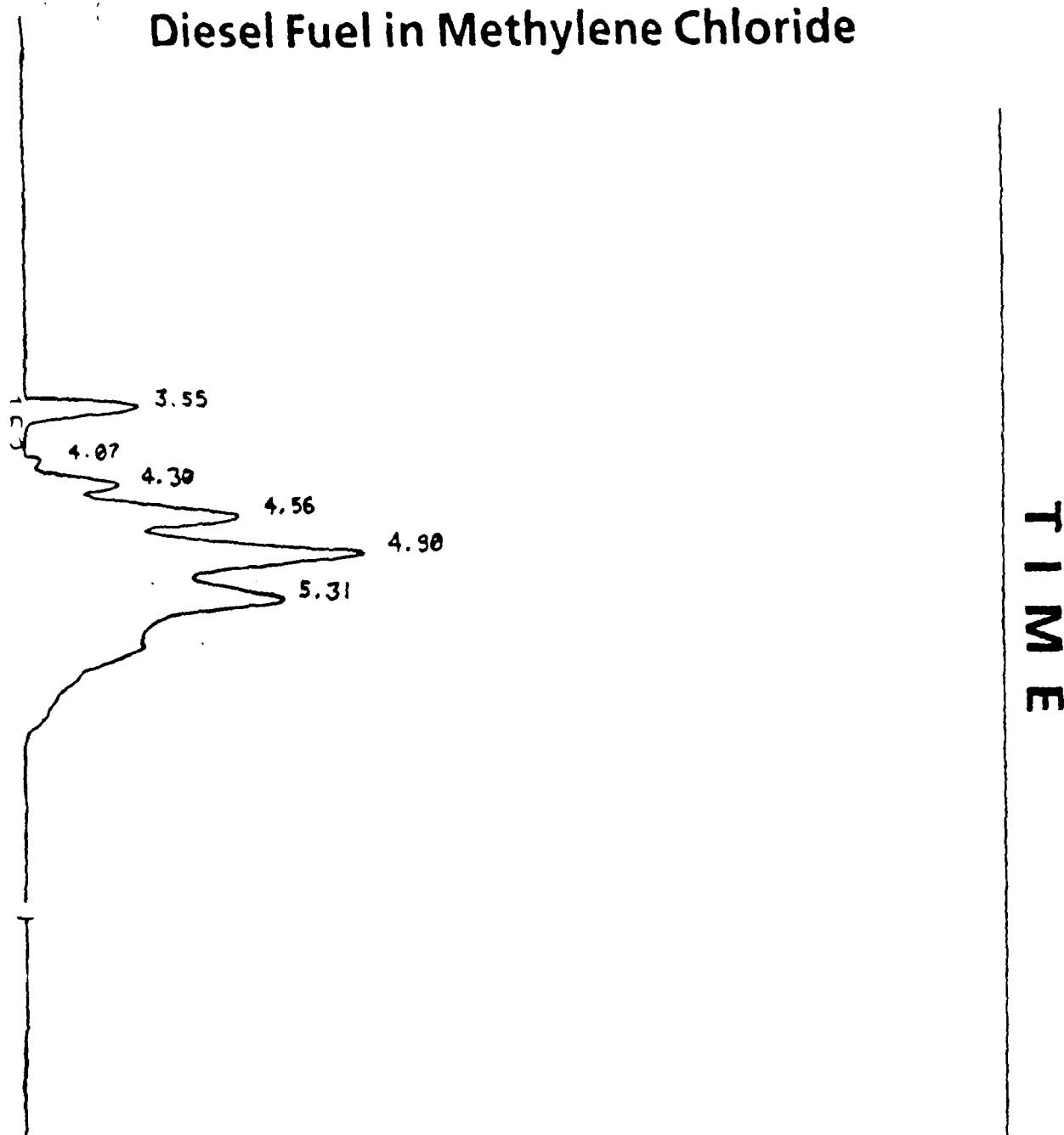
STANDARD QUANTIFICATION

AREA	AMOUNT	RT	AREA	RT
10111.0000	1.43	10111.0000	1.43	
10111.0000	1.55	10111.0000	1.55	
10111.0000		10111.0000		

Copy available to DTIC does not  
permit fully legible reproduction

## APPENDIX D

# Diesel Fuel in Methylene Chloride



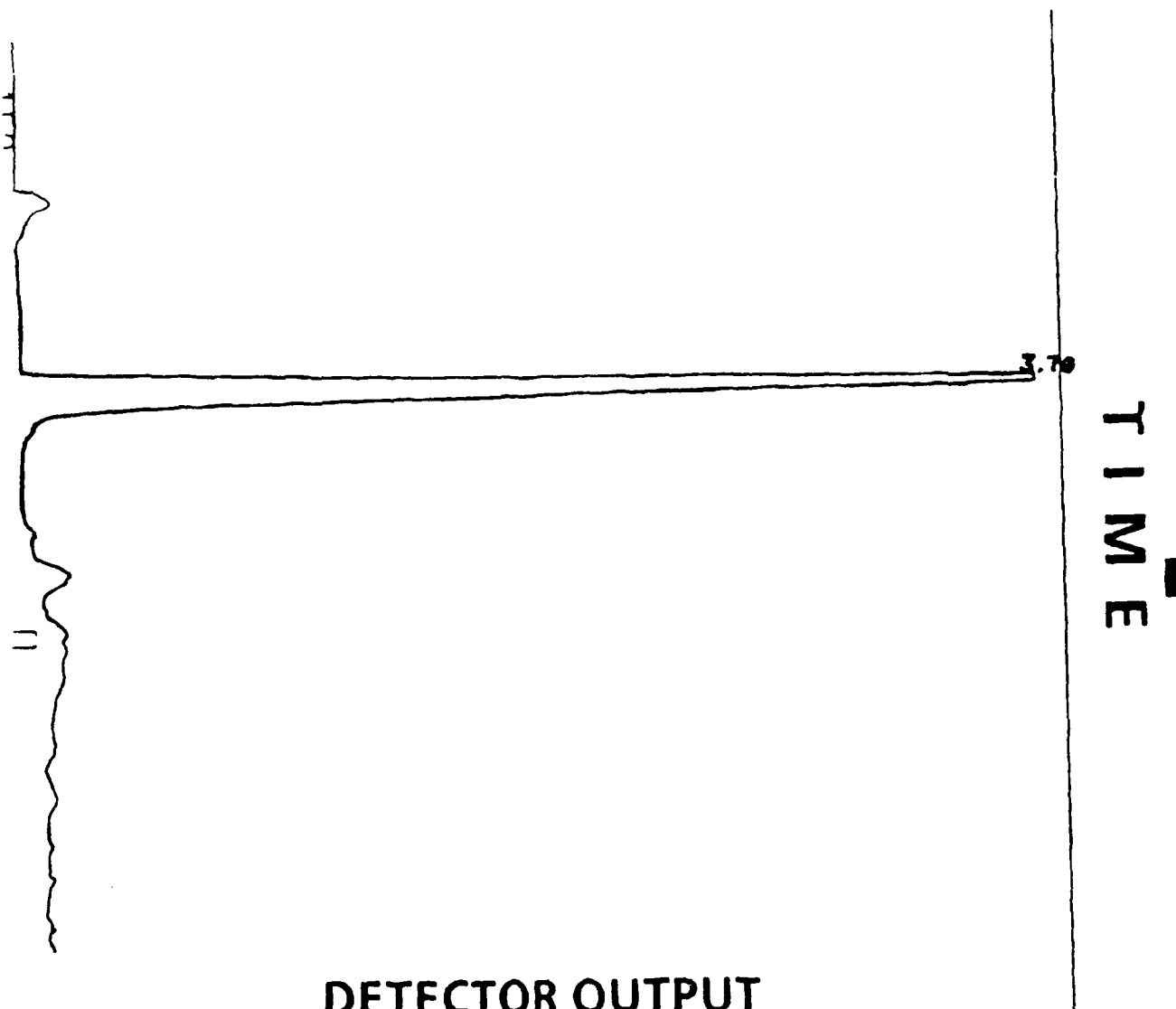
## DETECTOR OUTPUT

DETECTOR OUTPUT QUANTIFICATION

TIME	AMOUNT	PERCENT	PERCENT	PERCENT
3.55	1.00	1.00	1.00	1.00
4.07	1.00	1.00	1.00	1.00
4.30	1.00	1.00	1.00	1.00
4.56	1.00	1.00	1.00	1.00
4.90	1.00	1.00	1.00	1.00
5.31	1.00	1.00	1.00	1.00
TOTAL	6.00	6.00	6.00	6.00

Copy available to DTIC does not  
 permit fully legible reproduction

# HDO-30 Oil in Methylene Chloride

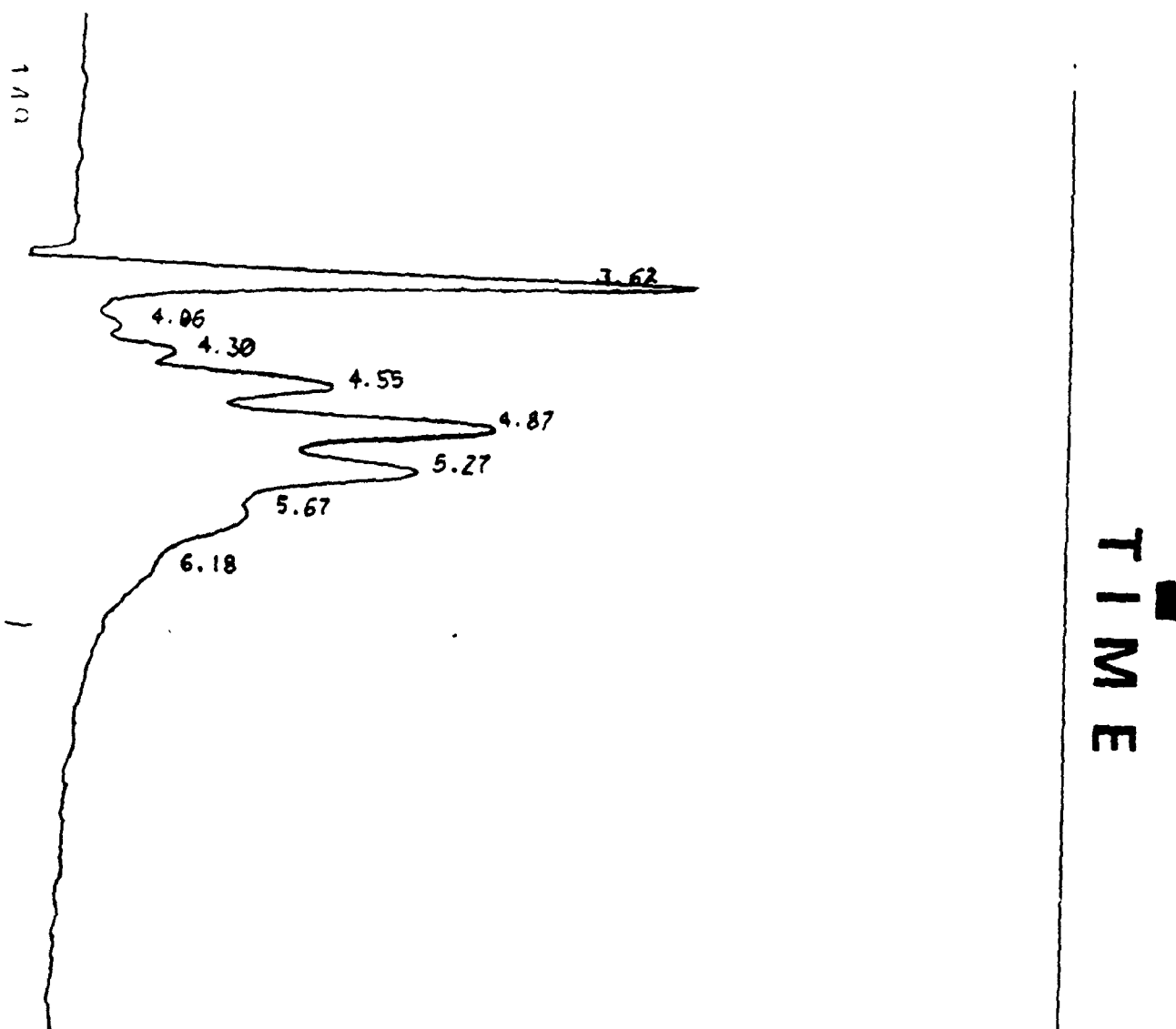


## DETECTOR OUTPUT

### EXTERNAL STANDARD QUANTITATION

PEAK#	AMOUNT	RT	EXP RT	AREA	RF
TOTAL	1449.81000	3.70		1449818 L	0.000000E0

# HDO-30 Crane Engine Oil in Methylene Chloride



## DETECTOR OUTPUT

STANDARD CONSTITUTION

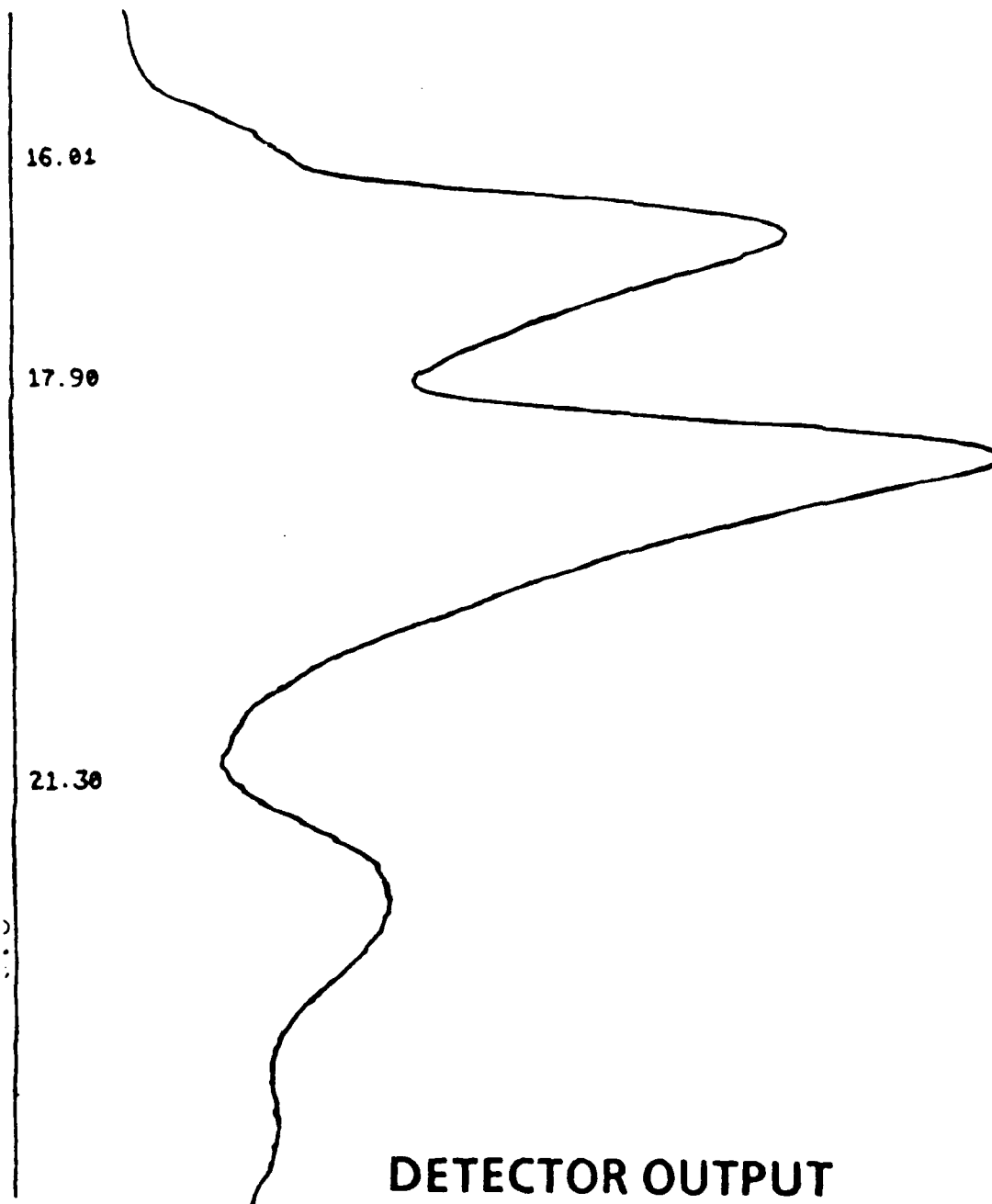
TIME	CONCENTRATION	TIME	CONCENTRATION
3.62	100.00000	4.06	100.00000
4.06	100.00000	4.30	100.00000
4.30	100.00000	4.55	100.00000
4.55	100.00000	4.87	100.00000
4.87	100.00000	5.27	100.00000
5.27	100.00000	5.67	100.00000
5.67	100.00000	6.18	100.00000
6.18	100.00000		

Copy available to DTIC does not  
permit fully legible reproduction

# **APPENDIX E**

# Naphthalene and Benzidine in Ethanol

TIME



## DETECTOR OUTPUT

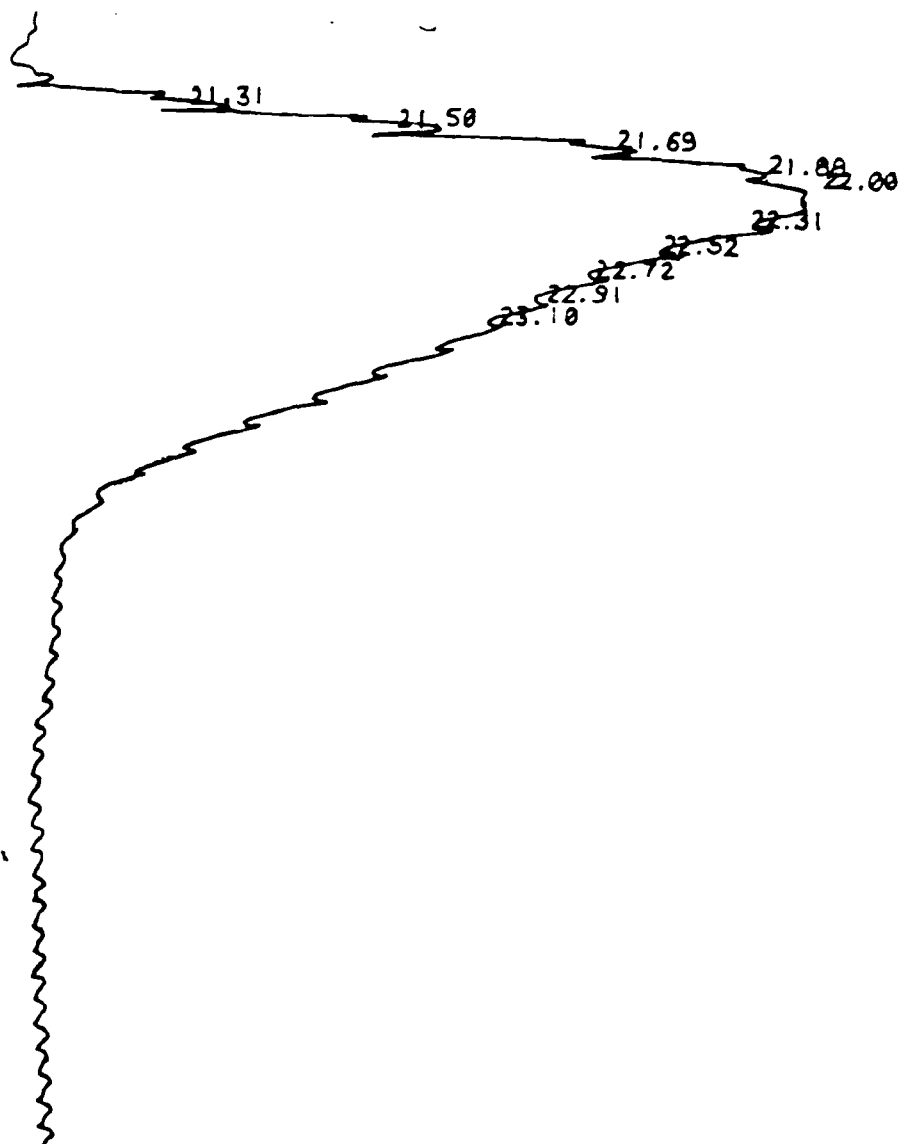
### EXTERNAL STANDARD QUANTITATION

PEAK#	AMOUNT	RT	EXP RT	AREA	RF
	3412.60000	16.01		3412619 L	0.00000000
	4726.82000	17.90		4726847 L	0.00000000
TOTAL	8139.42000				

## **APPENDIX F**



# Polypropylene Filter in Fuel



DETECTOR OUTPUT

TIME

STANDARD IDENTIFICATION

STANDARD

QUANTITY

ST

STANDARD

STANDARD

ST

1577.47000  
17534.50000  
15004.20000  
15149.00000  
13354.00000  
13353.40000  
15534.30000  
17917.00000  
1555.30000  
1752.14000

11.71  
11.73  
11.75  
11.77  
11.79  
11.81  
11.82  
11.83  
11.84  
11.85

1577534  
17535013  
15004542  
15149443  
13355737  
13353303  
15534563  
17917132  
1555342  
1752137

0.0000000  
0.0000000  
0.0000000  
0.0000000  
0.0000000  
0.0000000  
0.0000000  
0.0000000  
0.0000000  
0.0000000